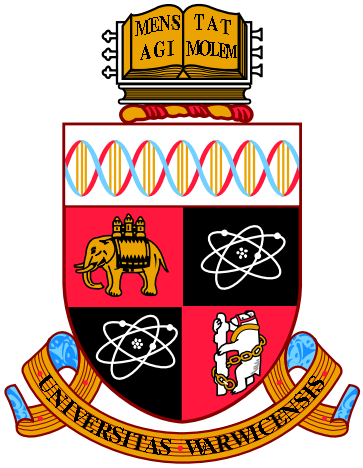


Efficient Streaming for High Fidelity Imaging

by

Joshua McNamee

Thesis

Submitted for the degree of **Doctor of Philosophy**.

WMG

September 2017

Abstract

Researchers and practitioners of graphics, visualisation and imaging have an ever-expanding list of technologies to account for, including (but not limited to) HDR, VR, 4K, 360°, light field and wide colour gamut. As these technologies move from theory to practice, the methods of encoding and transmitting this information need to become more advanced and capable year on year, placing greater demands on latency, bandwidth, and encoding performance.

High dynamic range (HDR) video is still in its infancy; the tools for capture, transmission and display of true HDR content are still restricted to professional technicians. Meanwhile, computer graphics are nowadays near-ubiquitous, but to achieve the highest fidelity in real or even reasonable time a user must be located at or near a supercomputer or other specialist workstation. These physical requirements mean that it is not always possible to demonstrate these graphics in any given place at any time, and when the graphics in question are intended to provide a virtual reality experience, the constrains on performance and latency are even tighter.

This thesis presents an overall framework for adapting upcoming imaging technologies for efficient streaming, constituting novel work across three areas of imaging technology. Over the course of the thesis, high dynamic range capture, transmission and display is considered, before specifically focusing on the transmission and display of high fidelity rendered graphics, including HDR graphics. Finally, this thesis considers the technical challenges posed by incoming head-mounted displays (HMDs). In addition, a full literature review is presented across all three of these areas, detailing state-of-the-art methods for approaching all three problem sets.

In the area of high dynamic range capture, transmission and display, a framework is presented and evaluated for efficient processing, streaming and encoding of high dynamic range video using general-purpose graphics processing unit (GPGPU) technologies.

For remote rendering, state-of-the-art methods of augmenting a streamed graphical render are adapted to incorporate HDR video and high fidelity graphics rendering, specifically with regards to path tracing.

Finally, a novel method is proposed for streaming graphics to a HMD for virtual reality (VR). This method utilises 360° projections to transmit and reproject stereo imagery to a HMD with minimal latency, with an adaptation for the rapid local production of depth maps.

Contents

List of Tables

List of Figures

Acknowledgements

The acknowledgements for this thesis begin with my supervisors, Prof. Alan Chalmers and Prof. Kurt Debattista, who never offered anything less than full confidence that it would happen.

To Becca, who stuck by me even in the three months where I could barely hold a regular conversation.

To my fellow candidates and close friends, Amar, Jon and Tim:

- Jon, I will never forgive you for getting me into this mess.
- Amar, I'm so sorry for getting you into this mess.
- Tim, thanks for driving us to Burger King so many times. All the best.

To my Mum and Dad, who didn't seem at all surprised when I said I needed to move back in.

To all the members of the research group: Ali, Carlo, Christopher, Debmayla, Demetris, Elmedin, Emmanuel, Martin (who owes me £100), Pinar, Ratnajit, Rossella, Sam, Stratos, Tom, Yuanfa, for their assistance and intelligence.

Thanks also to my friends and housemates from the last four years: Charlie Angelo, Sam Crawley, Stevie Finegan, Zoë Kelly, Jonathan Perkins and Alex Woolley, who all freely offered sympathy and support. And to Gerard McCaul, who did a little laugh and suggested I should have started writing up sooner.

Finally I would like to thank Prof. Stephen Butterfill, without whose twice unconditional kindness I would never have reached this point.

This work was made possible by generous funding from Jaguar Land Rover and the EPSRC.

Publications

The work in this thesis has contributed to the following publications:

- Josh McNamee, Kurt Debattista, and Alan Chalmers. Efficient Remote Rendering Using Equirectangular Projection. In Tao Ruan Wan and Franck Vidal, editors, *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2017. ISBN 978-3-03868-050-5. doi: 10.2312/cgvc.20171278

- J. McNamee, K. Debattista, and A. Chalmers. Remote rendering of high dynamic range graphics content. In *2016 Digital Media Industry Academic Forum (DMIAF)*, pages 6–10, July 2016. doi: 10.1109/DMIAF.2016.7574891

- Alan Chalmers, Joshua McNamee, Jonathan Hatchett, Ratnajit Mukherjee, Igor Olaizola, and Kurt Debattista. 12 bits is simply not enough for hdr video! *BEC, NAB*, 2015

- Joshua McNamee, Jonathan Hatchett, Kurt Debattista, and Alan Chalmers. Live hdr video streaming on commodity hardware. volume 9599, pages 9599 – 9599 – 11, 2015. doi: 10.1117/12.2187457. URL `http://dx.doi.org/10.1117/12.2187457`

- Joshua McNamee, Jon Hatchett, Kurt Debattista, and Alan Chalmers. Real time delivery of hdr video. *CVMP*, 2014 *(Poster)*

Note on the type

This thesis is set in the typeface Adobe Caslon Pro, produced by designer Carol Twombly based on William Caslon's papers of the 1700s.

Chapter 1

Introduction

> *Heracleitus is supposed to say that all things are in motion and nothing at rest; he compares them to the stream of a river, and says that you cannot go into the same water twice.*
>
> — Plato, *Cratylus*

Technology is — constantly in motion. The cutting edge of yesterday is quickly superseded by the cutting edge of today. New and advanced techniques and capabilities for high-fidelity imaging are emerging as computing power increases. Those concerned with image processing must now contend not only with HD resolution, but 4K resolution, high dynamic range (HDR), high frame rate (HFR), 360° video and stereo imagery, to name but a few high-profile examples. As well encroaching is the extension of imaging from standard display technology, where a rectangular screen sits at a distance from the user, to the now commercially-available head-mounted displays (HMDs) which are utilised to provide a virtual reality (VR) experience.

The accurate representation of visual reality is a long-promised goal of computing, forming a step change for human/computer interfaces. The nature of this goal, raising the level of fidelity across many distinct visual phenomena, requires that imaging technology be more realistic, more responsive, and more immersive than ever before. Both when capturing and representing imagery from the real world as well as when rendering imagery entirely within a computer, this means greater demands on resources and greater expectations of the technology provided by research to tackle this data.

This thesis is concerned with exploring this frontier within imaging, the interaction of the influx of new technologies and the corresponding demand on resources. The techniques approached in this thesis flow naturally from discussion of HDR video, to remote rendering, to virtual reality, with each of these layers adding more complexity to the images being represented and increasing the stress on the resources being used. It is possible to imagine an approach to this work, however, which proceeds through topic to topic in reverse, or in any order. These new challenges do not arrive neatly, in order, but all at once and must be met all at once.

1.1 Overview

The focus of this thesis is to assess and develop methods of streaming high-fidelity imaging to a remote location quickly enough that they can be interacted with in real-time, so that they can form part of a reactive, immersive virtual environment. The ability to stream content of almost any form is important because more often than not, the content we wish to view is not available to us locally. Online video streaming services have experienced significant growth over the last decade, to the point where they now form the majority of internet traffic [1]. Home streaming services permit the operation of interactive 3D games on a local computer or console from anywhere there is a client capable [2, 3]. There are still challenges to be met, though, in the streaming of high-fidelity imaging as an area of special interest.

1.1.1 High dynamic range video

The advent of HDR displays and HDR content moves display technology one step closer to being able to accurately represent the imagery of the outside world. The dynamic range of existing screens and video content has remained low, anywhere between 100 and 600 nits peak [4], for the entire history of computing, with only experimental research displays being able to present brighter content. The limited dynamic range of LDR screens restricts the content visible to well below the range of scenes commonly encountered in real-world lighting conditions [5].

To achieve the full potential of HDR video, it is crucial that all the dynamic range that was captured is delivered to the display device and tone mapping, the reduction of dynamic range, is confined only to the display. Furthermore, to ensure widespread uptake of HDR imaging, it should be low cost and available on commodity hardware. This thesis begins with an explora-

(a) This LDR image of a lake captures the sky and the baleful sun, but the detail in the town below is lost.

(b) This tone-mapped HDR image of a mountain valley captures both the sky and the shadows on the mountain side, but the tone-mapping results in an artificial visual effect.

Figure 1.1 – LDR and HDR Photographs.

tion of the best utility for general-purpose graphics processing unit (GPGPU) technology for delivering HDR end-to-end, from capture through transmission, storage and display.

HDR video is of interest in the context of remote rendering because most rendering engines by necessity process and output their graphics in a high dynamic range. This is essential because details such as reflections have to be able to take into account the quality of light in areas which are overexposed, for example, in order to calculate how the material which is reflecting the overexposed area should react. Transmitting HDR video and having the processing for display performed on the client, then is attractive for remote rendering because it expands the scope of operations which can be performed on the client by preserving more information in the transmission.

Investigating HDR video for remote rendering is thus a clear step within this research into expanding and improving upon existing methods of remote rendering.

1.1.2 Remote rendering

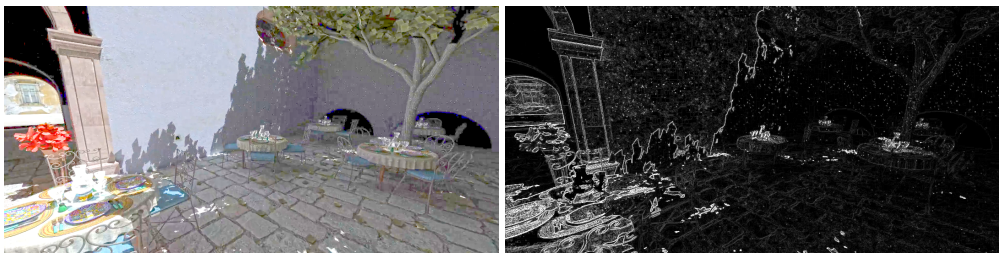
Computing power is rarely distributed how we would best wish; despite great advances in portable computing, some tasks, such as high-fidelity rendering, are still dependent on large mainframe computers and clusters. To achieve verisimilitude, we are forced to pick and choose from our methods of representing reality to fit the power of the device we have to hand. High-fidelity computer graphics have applications in many areas, from archeology, where the accurate passage of light through virtual scenes can give insight into the use of ancient buildings, through product design as well as recreation. The common factor is the desire to represent something which may not exist or else is physically impractical or expensive to experience.

What then is to be done when the item which is physically out of reach is the computing resource itself? Supercomputers and computing clusters possess physical sizes impractical for regular transportation, and expectations within computing are that Moore's Law, which states that the number of transistors capable of being packed in an integrated circuit doubles approximately every 2 years and has held since it was postulated in 1965, is likely cease to apply by 2025 [6].

A prospective solution to this is remote rendering, a resurrection of the mainframe idiom. A large, high-performance computer is used remotely, performing the bulk of the computation,

and a thin client in front of the user handles the presentation of the content and the return of user input. The cost of rendering computer graphics is even higher when considered in real-time terms. To provide a virtual environment which reacts to a user within it requires a degree of performance from the host system above and beyond that for so-called 'offline' rendering.

Encoding a stream of graphics as if it were regular video is a simple and practical solution to this problem already in use. It is however poorly adapted for graphics work, can be high cost in terms of resources and critically fails to take advantage specifically of the information available to the rendering engine in its encoding of video. This amounts to spending resources, which may be extremely expensive or scarce, for the renderer to perform actions which the video encoder discards.



(a) HDR image encoded at a low bit rate. (b) Signal-to-noise ratio map.

Figure 1.2 – This map of signal-to-noise ratio shows where detail has been lost in this low-quality encoding of a HDR source.

More advanced methods of taking data from computer graphics and transmitting it are still in their infancy. Remote rendering has the potential to enable graphics which were previously confined to the terminals of supercomputers to be transmitted with minimal bandwidth and minimal latency to a remote display on the level of a mobile phone. More than that, for a sufficiently large remote computer a device such as an Oculus Rift should be able to provide a virtual reality experience at a remote location with a level of realism previously only accessible in the physical vicinity of the computing resource.

This thesis investigates the means available for utilising the power of remote computers to improve the experience of remote users, and further explores the possibility of expanding this provision to HDR content.

Figure 1.3 – The Oculus Rift, a commercially available head-mounted display [18].

1.1.3 Virtual reality

VR requires the ability to generate visual environments with a high degree of accuracy, if sufficient authenticity is to be reached. For serious applications providing convincing virtual experiences to users, the high-fidelity graphics rendering requires the simulation of light transport using physically-based algorithms such as path tracing [7]. Being able to obtain these graphics in a timely manner, however, currently requires large, fixed computing resources. This is necessary if a head-mounted display (HMD) is to be used comfortably [8]. When these are not available locally, the solution is the use of remote rendering services, including those available in the cloud. Remote rendering permits resources like supercomputers or high-performance computing clusters to be used anywhere there is an internet connection and a client capable of receiving and displaying the data.

This accessibility however brings its own limitations. Efficient compression is required to transmit the rendered graphics from the remote rendering service to the user in a reasonable amount of time. A user wearing an HMD expects a minimal amount of latency between a movement and a response in the image, to avoid motion sickness [9]. In addition, a high level of quality for the content must be maintained to retain the verisimilitude of the rendered image; there is little point in using vast resources to display an image which is heavily compressed. These data requirements rapidly increase as stereo imagery to supply to both eyes is added.

Workstations, even relatively low-powered ones, today contain powerful local hardware which can be utilised for efficient division of work. Remote rendering always has a minimum cost to the client in terms of resources, even if it is only the cost of operating a hardware video decoder such as are commonly found in smart-phones. Client devices nowadays possess resources in excess of this which can be exploited to maximise the performance, in quality or latency, of a remote rendering system. A workstation, for example, will typically have a GPU capable of basic graphical rendering tasks. The application of this is not a straightforward problem, as the division needs to be simple enough that computing the combination of the two efforts does not negate the benefit gained compared to simply computing the whole render at the server.

1.2 Problem

Existing commercial methods of streaming graphics utilise software or hardware implementations of video codecs to stream the output from the rendering engine as a video stream. This can require a dedicated hardware implementation of a video codec as in commercial systems like NVIDIA NVENC [10], which are fast but tie you to a particular codec at a particular moment in time. If a hardware solution is not available, a software implementation is required, which will be necessarily costly in terms of resources like CPU time if it is to achieve high fidelity. Where the level of realism in a scene is bound by the resources of the system it is rendered on, this cost can be unacceptable or even prohibitive - especially in the case of emerging high-bandwidth technologies such as HDR and 2160p. 2160p resolution video requires four times as many pixels as 1080p video, and HDR video in floating point format requires four times as much data per pixel. Communication between a renderer and a specifically-developed encoding scheme can reap benefits from additional information available in the course of rendering, mitigating the cost of encoding this large amount of data by traditional methods.

Methods for optimising streaming video for remote rendering have been proposed, as well as means of augmenting a video stream using the data available to a rendering engine. The existing methods however are specific in scope and there is no consensus on which is most useful or best for any given scenario.

1.3 Research objectives

The initial goals set for this thesis are thus:

- To develop a framework through which HDR video can be analysed and processed in real-time.
- To develop a novel adaptation of existing remote rendering techniques to HDR video.
- To develop a novel approach to streaming graphics to a HMD through reprojecting 360° spherical video, in both LDR and HDR.

1.4 Outline

This thesis is organised as follows:

- Chapter 2 concerns the relevant background material for this thesis in rendering.
- Chapter 3 completes the survey of work prior to this thesis in the areas of remote rendering and virtual reality.
- Chapter 4 consists of an exposition of HDR imaging and associated techniques, specifically discussing HDR video.
- Chapter 5 sets out in context the research which this thesis has undertaken, and presents the research question.
- Chapter 6 describes the implementation and evaluation of a full pipeline for end-to-end HDR video capture and display.
- Chapter 7 presents a novel adaptation of remote rendering to high-fidelity graphics and assessment thereof.
- Chapter 8 proposes a system for streaming low-latency virtual reality via the local reprojection of equirectangular 360° maps, then extends it to HDR streaming.
- Chapter 9 is a summary and conclusion of the goals, achieved and planned, of this thesis and a sketch for continued research in these areas.

Chapter 2

Rendering

> *You must judge that the rays of this light are nothing other than the lines along which this action tends. So that there are an infinite number of such rays which come from all the points of the luminous bodies towards all the points of the bodies that they illuminate.*
>
> — René Descartes, *La Dioptrique*

High fidelity imaging encompasses many fields of research, including real-world image capture techniques such as camera technology, as well as high-fidelity rendering methods.

This chapter will present the relevant background material, to define the terms for the chapters to come. The material is split into chapters on rendering, HDR techniques, and remote rendering techniques.

First presented is a firm base of rendering concepts which further chapters will depend on. This includes a summary of the physical basis of graphics rendering in terms of light and progresses onto more specific areas of computer graphics which will be relevant in further chapters, specifically general-purpose GPU, distributed ray tracing and then on to HDR material.

2.1 Rendering

We use the term rendering to mean the process of using a computer and/or an algorithm to turn data and method into virtual experience. Specifically, we use rendering within the context of computer graphics to mean the process of using a computer with or without dedicated graphics hardware to produce a realistic, or otherwise, depiction of a scene represented by a set of data.

2.1.1 Radiometry

Radiometry is the study of light energy. For the computation of rendering which simulates light as it acts in real-world scenes, an understanding of radiometry is thus essential. Sequential to this is the study of photometry, the effect of light on the Human Visual System (HVS). Taking the two techniques together, the simulation of physical light and the simulation of how light is perceived by human beings, allows us to consider high-quality graphical rendering.

Within radiometry are several quantities directly relevant to our work in graphics rendering [20]. Radiant power or flux is measured in Watts (Joules/second) and represents an amount of light energy moving (emitting or being received) independent of the size or direction of the source.

$$\textit{Radiant Power} := \Phi \tag{2.1}$$

From radiant power we can define two more quantities, irradiance and radiosity. Irradiance denotes the radiant power received by a surface in terms of area in Watts per metre squared.

$$\textit{Irradiance} := E = \frac{d\Phi_{in}}{dA} \tag{2.2}$$

And conversely radiosity (sometimes termed 'radiant exitance') denotes the radiant power emitted by a surface in terms of area, again in Watts per metre squared.

$$\textit{Radiosity} := B = \frac{d\Phi_{out}}{dA} \tag{2.3}$$

Finally, and most pertinent to our objective of simulating the view of an eye in a scene, the radiance is a quantity defined as the radiant power per second, per square metre, per solid unit angle (the unit angle is the radian^2, or steradian). Simply put, the amount of light coming off or being received by an area object from or in a direction.

Radiance varies in five dimensions and is typically expressed in terms of point x and directional vector Θ.

$$\textit{Radiance} := L(x, \Theta) = \frac{d^2\Phi}{d\omega dA^{\perp}} = \frac{d^2\Phi}{d\omega dAcos\theta} \tag{2.4}$$

From these definitions, the relationships between these quantities can be expressed as integrals:

$$\Phi = \int_A \int_\Omega L(x, \Theta_{out}) cos\theta d\omega dA_x \tag{2.5}$$

$$E(x) = \int_\Omega L(x, \Theta_{in}) cos\theta d\omega_\Theta \tag{2.6}$$

$$B(x) = \int_\Omega L(x, \Theta_{out}) cos\theta d\omega_\Theta \tag{2.7}$$

Where Ω is the total solid angle. Also worth introducing is an intuitive notation for radiance: $L(x \rightarrow \theta)$ indicates the light leaving a point x in a direction θ and $L(x \leftarrow \theta)$ indicates the light arriving at a point x from a direction θ.

2.1.2 The rendering equation

In a 1986 paper, Kajiya et al. [21] describe an equation for modelling light scattering off various kinds of surfaces, based on similar equations applicable to radiative heat transfer. The equation they present, adapted to our term L, is as follows:

$$L(x, \theta) = g(x, x')[\epsilon(x, x') + \int_S \rho(x, x', x'') L(x', \theta') dx''] \tag{2.8}$$

Where x, x' and x'' are points in S, the union of all available surfaces in the scene. θ and θ' are directions defined by the path between points x and x' and x' and x''. A function g codes for occlusion of the path. ϵ is a term representing the emitted light from x to x' and ρ is a term representing light scattered by x' to x from all points x''.

The rendering equation expresses that the light transferred to a point from another location is the sum of the light emitted by that location and the light reflected by that location in the direction of the original point.

$$L(x \rightarrow \theta) = L_e(x \rightarrow \theta) + L_r(x \rightarrow \theta) \tag{2.9}$$

L_e represents the light emitted and L_r represents the light reflected by the location [20].

2.1.3 The interaction of light with a surface

Solving the rendering equation is equivalent to having an algorithm for rendering a virtual scene, and the accuracy of the solution informs the accuracy of the rendering. In order to get there however, we are in need of means of solving the components within it. While the properties of emitted light L_e can be easily quantified (i.e. with a light meter) we are in need of a means of calculating with some degree of accuracy the manner in which light is reflected and refracted by a surface, to form our term L_r.

Nicodemus [22] denotes a function named a bidirectional scattering-surface reflectance-distribution function (BSSRDF) to be a single function accounting for the geometrical reflectance properties of a surface. For most cases a simpler function, the bidirectional reflectance-distribution function (BRDF) suffices. It operates with the assumption that light is incident and reflected at the same point, ignoring sub-surface scattering (such as you might see on a marble counter-top). A BRDF describes the ratio between the reflected differential radiance in a direction θ and the differential irradiance incident through a differential angle $d\omega_\Psi$ at a point x with a normal N_x.

$$f_r(x, \Psi \rightarrow \Theta) = \frac{dL_r(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)} = \frac{dL_r(x \rightarrow \Theta)}{L(x \leftarrow \Psi)cos(N_x, \Psi)d\omega_\Psi} \tag{2.10}$$

The function acquires the same value regardless of which of the two angles is incident and which is exitant. This property, *Helmholtz Reciprocity*, is denoted $f_r(x, \Psi \leftrightarrow \Theta)$.

2.1.3.1 Bidirectional Reflectance Distribution Functions and the Rendering Equation

Rearranging Equation 2.10, we can produce an equation for $L_r(x \rightarrow \Theta)$.

$$L_r(x \rightarrow \Theta) = \int_{\Omega_x} f_r(x, \Theta \rightarrow \Psi)L(x \leftarrow \Psi)cos(N_x, \Psi)d\omega_\Psi \tag{2.11}$$

Which allows us to give a solution to the rendering equation for BRDFs:

$$L(x \rightarrow \theta) = L_e(x \rightarrow \theta) + \int_{\Omega_x} f_r(x, \Theta \rightarrow \Psi)L(x \leftarrow \Psi)cos(N_x, \Psi)d\omega_\Psi \tag{2.12}$$

This equation is a second degree Fredholm integral - it cannot be solved analytically as radiance appears as a term recursively. It can be solved numerically either by considering finite elements or through point sampling methods, as will be discussed in the following sections.

2.1.4 Ray tracing

Ray tracing is an intuitive means of solving the rendering equation through sampling, either regularly or otherwise. There is a major stumbling block in simulating light as it actually occurs (i.e. from a light source until it hits the eye), namely that many or most rays from a light source are absorbed by objects other than an eye. Ray tracing is often formulated such that it shoots rays from the eye back until they reach a light source in order to avoid this burden. It is a concept considerably older than that of computer graphics, considered (although not originally!) by René Descartes with regard to the creation of rainbows [19].

Ray tracing simulates the transportation of light around a scene by taking rays out from a camera or eye to an object, and then from that object by reflection or refraction in one or many directions, repeating the process with further rays, repeatedly calculating intersections between rays and objects until a light source is reached or the engine which is tracing the ray is by some other metric satisfied. In the categories used by Shirley et al. [23], it is an 'image-order' rendering method as it is concerned with the relationship between the pixels which make up the image and the rays it traces. This is not to say that it may not be performance-restricted by the quantity or quality of objects in the scene, however.

The first example of ray tracing in computer graphics is generally attributed to Appel [24], who proposed shading computer generated imagery by shooting a single ray for each pixel in order to detect object depth order and distance. The process of firing a ray or rays plural for each pixel in an image rendering is at first called ray casting. If it progresses to shooting a tree of rays to determine a path of light further than the first object, it is termed ray tracing [25]. Tracing rays recursively was first coined by Whitted [26], who proposed simulating the interaction of light with object surfaces by shooting reflected or transmitted rays recursively, as well as shooting rays towards light sources to detect shadowing.

A ray tracer is typically broken down into the generation of rays, the detection of intersections and finally the computation of output colour based on the rays.

2.1.5 Rasterisation

Rasterisation, or scanline rendering as it is also known, is the methods of graphics rendering typically used by computer games and most operations which requires real-time interactivity between a user and a graphical environment. Rasterisation operates in order of objects within a scene, drawing each object into its correct place in turn and a separate process lights the resulting projected scene *ex post facto*.

The popularity of rasterisation is derived in part from the preponderance of dedicated hardware for rasterisation engines in the form of the GPU, as well as free-to-use graphics libraries such as OpenGL [27] and DirectX which provide access to this hardware in a standardised way.

The graphics pipeline, as it is called, for rasterisation can contain many stages or relatively few, but typically it will begin with an application which generates the scene according to the state of an internal engine or else generate the scene from a description file. Once the scene is generated and loaded into graphics memory as a set of primitives (usually 3D triangle vectors), these primitives are operated on by a program which may perform operations per-vertex such as lighting or clipping called a vertex shader. Following modifications to the geometry, the visible surfaces are calculated and per-pixel operations are performed such as pixel shading. This is a very broad overview and a vast number of methods for shading in a rasterisation engine have been produced [28].

2.1.6 General purpose GPU

While it may have been possible just a few years ago to state that graphics hardware is able to accelerate rasterisation engines but not ray tracing engines or similar, modern graphics hardware from all three major graphics manufacturers supports general-purpose GPU (GPGPU) work. Languages such as CUDA [29] and OpenCL [30], as well as optimisation libraries like C++ AMP [31] allow generic code of any intent to be run in parallel across anywhere from a handful processing units, to several thousand. This has lead to the development of ray tracing engines which are designed directly to capitalise on this capacity, with varying results[32].

2.1.7 Distributed ray tracing

Distributed ray tracing was proposed by Cook et al [33], who characterised the traditional ray tracing algorithm as excessively rigid. By replacing the fixed generated rays with rays distributed according to the function they were sampling, they claimed, it was possible to achieve lighting effects ray tracing had heretofore not been able to recreate as a matter of course such as motion blur, depth of field, translucency, and/or fuzzy reflections. The algorithm depends on stochastic sampling, using Monte Carlo integration to estimate a solution to the rendering engine.

2.1.7.1 Monte Carlo

Monte Carlo methods estimate the solution to an integral by repeatedly sampling it (M times) according to a probability function ($p(x)$) and averaging the result. Given a generic integral I over a domain D:

$$I = \int_D f(x)dx \tag{2.13}$$

We define $\langle I \rangle$, our Monte Carlo estimator:

$$\langle I \rangle = \frac{1}{M} \sum_{i=1}^{M} \frac{f(x_i)}{p(x_i)} \tag{2.14}$$

2.1.7.2 Monte Carlo and the Rendering Equation

We can apply this to our equation for reflected radiance (Equation 2.10) to estimate a solution to that integral:

$$\langle L_r(x \rightarrow \Theta) \rangle = \frac{1}{M} \sum_{i=1}^{M} \frac{f_r(x, \Theta \leftrightarrow \Psi_i) L(x \leftarrow \Psi_i) cos(N_x, \Psi_i)}{p(\Psi_i)} \tag{2.15}$$

To calculate the radiance arriving at the point, $L(x \leftarrow \Psi_i)$, more rays are fired to simulate the light reflecting and/or refracting to the rest of the scene. On top of that, shadow rays must be fired towards ray sources. This can rapidly cause the rays to multiply beyond count, so a means of causing the rays to expire must be used.

The fuzzy-generated rays which permit soft shadows and the like in distributed ray tracing also introduce high frequency noise unless a great many samples are taken, as the value of any

two pixels is decided without any spatial coherence. Unless large amounts of samples are taken, a Monte Carlo generated image will look noisy.

2.2 Upsampling and graphics

High-quality graphics rendering is an expensive process. Access to machines, time running machines, power to run machines are all metered resources, and even the most efficient algorithms can have high requirements if fidelity is required.

Working with graphics in this environment is a difficult, time-consuming process. There are many scenarios in which being able to acquire an acceptably inaccurate final image quickly is preferable to acquiring a correct image at an unacceptable time cost. In an output medium like fast-moving video as used in movies, being able to quickly render a scene which gives a feel for the movement of a camera around an object and the layout of colour on the screen is inherently valuable, for instance. Methods of acceleration which cut down on render time at an acceptable cost of accuracy, or else converge in the absence of being removed from the camera view to accuracy over time, are thus of interest.

The cost of many aspects of rendering accurate graphics can be measured per-pixel, e.g. many methods of lighting. Reducing the number of pixels which need to be processed is thus an efficient method of reducing the cost of rendering. Pixels can be estimated both in space, as close regions are likely to be lit similarly, and time, as objects retain much of the same lighting as they move across a field of view.

2.2.1 Spatial upsampling

Spatial upsampling or upscaling is the process of taking a single static image and increasing it in resolution based on the data and metadata for that image.

Most scenes commonly rendered feature some degree of spatial coherence. That is, regions located close to each other are more likely to be the same colour. This has obvious real-world parallels, such as painted walls or coloured materials. Spatial coherence is thus an obvious pick for efficient rendering, as if we can guess the colour of a pixel reliably based on the pixels surrounding it we do not need to draw ever pixel 'blind'.

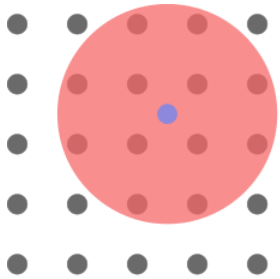


Figure 2.1 – k-nearest pixels (in this case $k = 10$) are within the read circle surrounding our blue pixel in the upscaled image.

Nearest-neighbour

Nearest-neighbour upsampling is a common, straightforward method of increasing an image's resolution. Each pixel in the upsampled image is set according to the pixel it is closest to when the larger image is projected onto the smaller one. It gives a blocky result, as the aliasing in the smaller image is magnified.

k-means upscaling

Rather than taking the nearest estimate of the value of a pixel, alternative results can be gotten by instead taking a weighted average of k pixels surrounding the one we wish to value, as shown in Figure 2.1. The method proposed by Shepard [34] weights the averages according to distance from the pixel being computed, as the accuracy of the pixel ought to decrease with distance.

Many other methods have been proposed of varying complexity, but they are often computationally intensive for the results they give.

Gaussian filtering

Gaussian filtering achieves a blur over the image by operating a gaussian kernel, an averaging of the value of pixels over a region per-pixel. A gaussian filter can be used to smooth an upsampled image, but it introduces a significant blur and does not preserve hard edges.

Bilateral filtering

Tomasi and Manduchi [35] proposed a filter following the gaussian filter which would

smooth an image while maintaining detected edges. It does this by using a range filter as well as the spatial kernel to weight values which deviate less. For a position p and a result J_p:

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega} I_q f(\| p - q \|) g(\| I_p - I_q \|) \tag{2.16}$$

Where f is the spatial filter on positions p, q and g is the range filter on the corresponding image values I_p, I_q. k_p is a normalising value and Ω is the set of values we cover with the kernel.

High-dimensional Gaussian filtering

While directly stretching a low resolution image to a higher resolution is something of a blunt instrument, more subtle methods can be utilised to guide a higher resolution image created from low resolution information. Specifically, high resolution metadata buffers can guide edge-aware methods like bilateral filters. A high resolution depth buffer, geometry buffer or colour map can provide distinct edges to what would be an otherwise blocky upsampling.

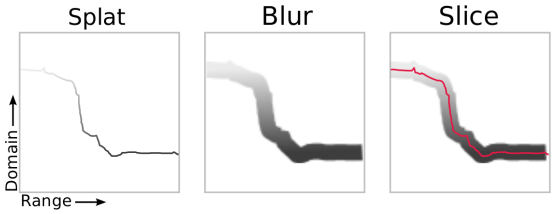


Figure 2.2 – The splat-blur-slice pattern. The red line is the re-sampled points.

Adams et al. [36] detail a generalised high-dimensional Gaussian filter using the permutohedral lattice, a reimplementation of the splat-blur-slice model proposed in [37]. In that paper, the authors propose that the process of computing Gaussian-like filters can be achieved quickly by embedding the input signal in a high-dimensional space (splatting), performing an unguided gaussian blur on that space (blurring) and then retrieving the original signal as samples from the now-blurred space (slicing). The splat-blur-slice pattern is illustrated in Figure 2.2.

The permutohedral lattice places the signal points it receives into a high-dimensional lattice constructed from uniform simplices, permitting computationally simple barycentric interpolation in both the splatting stage as well as simple access to local neighbour simplices in the blurring stage.

With a high-dimensional filter in the manner of a bilateral filter, it is possible to use the guidance of one or more metadata buffers in such a lattice to guide the edges of an upsampled and filtered low-resolution full render. This includes depth buffers, geometry buffers, motion buffers and potentially more specific buffers such as reflection edges. Both [38] and [39] which we will cover shortly include guided high-dimension filtering as part of a larger caching strategy.

Joint bilateral filtering

Kopf et al. [40] adapted bilateral filtering in this fashion to describe Joint Bilateral Upsampling, a method for combining a low-resolution set of data extracted from an image with the edge data in the original high-resolution in order to upsample the data.

$$\tilde{S}_p = \frac{1}{k_p} \sum_{q\downarrow \in \Omega} I_q f(\parallel p\downarrow - q\downarrow \parallel) g(\parallel \tilde{I}_p - \tilde{I}_q \parallel) \tag{2.17}$$

Where p and q are the image coordinates, $\tilde{I}$ represents the original image, S represents the downsampled image, $\tilde{S}$ represents the upsampled output, and the $\downarrow$ indicates coordinates which are mapped to the low-resolution image. f and g remain as filter weights, and k_p remains as a normalising factor. This method was adapted by Yang et al. [41] to rendering, demonstrating the use of a high-resolution geometry buffer as a set of edges to guide an upsampling of a low-resolution full render.

2.2.2 Temporal caching

In Nehab et al. [42], the authors proposed a system of estimating motion in pixel-space, such that a buffer could be generated of per-pixel coordinates pointing to the location in previous frames of the same spot being shaded for the current pixel. In this way, the number of new pixels which need shading in a single frame render can in theory be reduced to merely those

which are disoccluded by camera and object motion. In practise, the authors found that any scene involving non-trivial lighting requires regular sampling so that the rendered frames can 'converge' on an accurate rendering of the scene.

The paper calls this 'Reverse Reprojection Caching' due to the method by which the objects in view of the camera are projected back to the previous camera position. Here, we will refer to it as 'Temporal Caching' to highlight its nature as reaching back through time as opposed to space.

2.2.2.1 Determining coordinates

For platforms leveraging hardware graphics support, Nehab et al. [42] recommend computing the reprojection coordinates of each vertex and relying on hardware interpolation support to assign per-pixel coordinates.

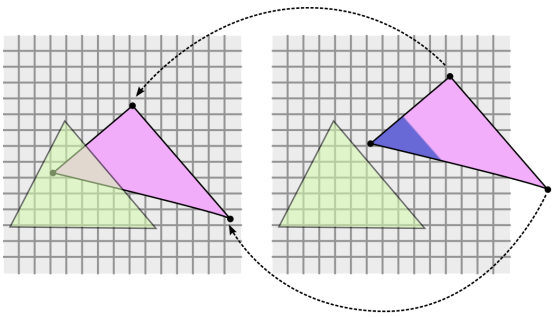


Figure 2.3 – Vertices in the second frame are reprojected back into the previous one. The area shaded blue is disoccluded and thus cannot be retrieved from the temporal cache.

More generally, at time t we take our pixel position $p_t = (i_t, j_t)$, use the camera matrix N_t to locate the position in the scene (x_t, y_t, z_t, w_t), apply the inverse of any scene motion M_t^{-1} to obtain $(x_{t-1}, y_{t-1}, z_{t-1}, w_{t-1})$, the coordinates of the point in the scene at time $t-1$, and finally apply the inverse camera matrix from time $t-1$, N_{t-1}^{-1}, to obtain p_{t-1}, the coordinates in the previous screen buffer which ought to match the pixel in the current scene. From this, we store in a screen-sized 'motion buffer' the coordinate differences:

$$m_t = p_t - p_{t-1} = (i_t - N_{t-1}^{-1} M_t^{-1} N i_t, j_t - N_{t-1}^{-1} M_t^{-1} N j_t) \tag{2.18}$$

Of course, if objects or the camera move in such a way that the pixel at $p_{t-1} = (i_{t-1}, j_{t-1})$

is sampling a closer (or farther) object than the object in view at $p_t = (i_t, j_t)$ then the cache hit will be invalid. The blue region seen in Figure 2.3 meets this case.

2.2.2.2 Detecting misses

Misses are checked for by storing the calculated depth when we take the pixel at time $t - 1$ along with the coordinate motion. This depth can then be checked against the previous frame depth buffer to ensure that the object in view in the previous frame is the same as the object in view in the current frame at the two pixel locations.

2.2.2.3 Efficient processing

An assessment of the most efficient implementations of these methods has been published in Sitthi-amorn et al. [43], which considers various structures of algorithm in the context of the graphics pipeline.

Single pass The method proposed in Nehab et al. [42] can operate entirely within a single pass of a shader, using motion vectors to move the vertices and allowing a fragment shader to check whether or not the cache can be hit for the per-pixel values, and either use the cache or calculate the shading on that pixel from scratch accordingly. As both Nehab et al. [42] and Sitthi-amorn et al. [43] note however, this is very inefficient for branch prediction. In the worst case scenario, a particle passing by a stationary object could cause a full processing unit of cache hits except for one run of the shader program, almost entirely negating the benefits.

Two pass To account for this, Nehab et al. [42] offers a second shading schema requiring two passes over the buffer in which the first either hits the cache or else stores a signal value which informs the pipeline that only the remaining pixels require the full shading program to be run. While this method is preferable to the first method in that it does not delay blocks of execution which could rely mostly on the cache in order to shade select pixels fully, it is still vulnerable to blocks mostly composed of the shade-fully instruction being delayed by the complexity of the cache hit algorithm.

Three pass Thusly, Sitthi-amorn [43] proposes a three-pass shading method for the cache in which rather than running the full shader program for each pixel which is not covered by the cache, instead a second stage after the cache hit check precomputes refreshed values directly into the cache rather than to the output buffer. In this way, while three passes are required over a buffer each pass has predictable timing across the image. While this approach can have negative effect where the cost of passing over the data three times can cancel out the benefit gained or where decoupling the full shading from the precomputation for the cache breaks compiler optimisations, the paper suggests that the scenarios where these issues are prominent may be unsuited to temporal caching entirely.

2.2.2.4 Refreshing

Critical to the performance of the temporal caching is the pattern of refreshing used. Nehab et al. [42] conclude that randomly sampling $\frac{1}{\Delta n}$% of pixels per frame will on average refresh the entire cache every Δn frames, and that this is preferable to refreshing the cache in contiguous blocks, i.e. dividing the frame into Δn blocks and refreshing each of those in turn. This random sampling avoids visible lines seeping into the image where the cache has lost coherency.

In Sitthi-amorn et al. [43] the authors concur with these findings except to suggest that the random samples be (for example) 4x4 blocks to aid branch prediction. They demonstrate a small to large performance improvement depending on the number of passes for using 4x4 blocks.

2.2.3 Spatio-temporal caching

A combination of the two discussed methods, spatial and temporal upsampling, is an obvious choice. Ideally we would like to see the benefits of spatial upsampling, namely the speed and minimal dependence on data other than that which is displayed with the advantages of temporal upsampling, combined with the convergence to a 'perfect' image which features in temporal caching. Described as 'Spatio-temporal Upsampling' in Herzog et al. [38], the paper proposes a method for combining the two based on depth and geometry aware weighting, as well as combining the temporal 'history' of the frame into a single buffer.

The idealised equation for output pixel h can be expressed as:

$$h(i) = \frac{1}{\sum w_s w_t w_f} \sum_{q=0}^{T} \sum_{j \in Ni(t-q)} w_s(i(t-q), j) w_t(t-q, j) w_f(q) l(\hat{j}, t-q) \tag{2.19}$$

Composed of a spatial term w_s and a temporal term w_t weighting a low-resolution pixel l, taken over a neighbourhood N surrounding the pixel being processed with a temporal history of T previous frames indexed q. A term w_f acts to weight older cache values, reducing their influence.

This is of course a highly impractical equation to try and implement. It requires several potentially complex weights to be calculated several times per pixel per time step. Herzog et al. [38] detail a method of combining the bulk of these lookups into a single rolling buffer, which we will consider.

The spatial weight w_s is defined as a weight on the per-pixel geometry normals $\bar{n}_i$ and the per-pixel depth values z_i. σ_n is chosen as 0.2 to give a reasonably strict window for geometric variation and σ_z is chosen as a few percent of the overall depth range of the scene. The paper suggests 3% of the difference between the near and far frustum plane.

$$w_s(i,j) = g(max(0, 1 - (\bar{n}_i \bullet \bar{n}_j))^2, \sigma_n^2) \cdot g(|z_i - z_j|^2, \sigma_z^2) \cdot k(i,j) \tag{2.20}$$

$$k(i,j) := g(||x(i) - x(j)||, r, \gamma); \tag{2.21}$$

$$g(x, \sigma, \gamma) := (max(\epsilon, 1 - \frac{x}{\sigma}))^{\gamma} \tag{2.22}$$

Herzog et al. use $\gamma = 3$ for all cases of g used. The temporal weight w_t is a binary control on whether or not the pixel was drawn at the time t in question. The weight w_f can be an exponential fall-off.

$$w_f(q) := 0.9^q \tag{2.23}$$

2.2.3.1 Temporal refreshing and jitter

Herzog et al. [38] also describe a method of refreshing the temporal cache by regularly sampling the full shaded output in 4x4 subsampled blocks. By targeting the refreshed pixels in a sequence within this block, both the temporal cache can be continuously refreshed and the low-resolution

image for spatial upsampling can be generated at the same time. Care must be taken to appropriately apply weights to the pixels which are jittered in this way, to avoid the image appearing to shimmer around the edges.

The pattern chosen, as seen in Figure 2.4 is an easy-to-compute fill of a 4x4 block, allowing it to be generated without concern for the performance cost of the random samples. Aliasing may be an issue. Due to the regular nature of the pattern, branch prediction is easily maintained by only processing each 4x4 block of pixels once, for the selected pixel within that. As there is no branching to occur, branch prediction is not a worry.

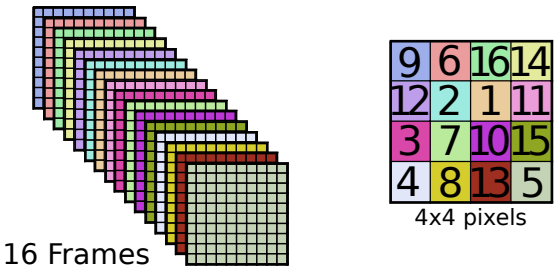


Figure 2.4 – A regular sampling pattern for temporal jittering, as used in Herzog et al [38].

2.2.3.2 History buffer

To avoid having to store and cycle T previous upsampled frames as inputs to the spatio-temporal upsampler, Herzog et al. [38] propose a history buffer which caches the previous upsampled result and the previous upsampled weighting as a pre-processed input for the most recent frame. This permits the construction of an equation which fulfils the same purpose but has no temporal dependencies.

$$h(i) = \frac{\bar{h}(i)\bar{w}(i) + w_f(1)h_w(i(t-1))h_r(i(t-1)}{\bar{w}(i) + w_f(1)h_w(i(t-1))} \tag{2.24}$$

h_r is the pixel result from the previous frame and h_w is the pixel weight from the previous frame. The buffer is updated after the frame is rendered with $h(i)$ replacing $h_r(i)$ and $h_w(i)$ being defined as:

$$h_w(i) := \bar{w}(i) + w_f(1)h_w(i(t-1)) \tag{2.25}$$

2.3 Depth compression with video encoding

Depth maps are not patterned like a regular image. They consists mainly of smooth gradient and hard edges. Many methods have been proposed for the efficient streaming of depth information alongside an image, often depending on shared information between the frames. To this end, a specialised encoding method is justified and will achieve better compression results than the transmission of this data as a regular video stream[44]. Multi-view coding plus depth (MVD)[45] is capable of combining an image frame alongside a depth frame using inter-view prediction as well as standard intra-frame prediction.

2.4 Rendering 360° Projections

In order to encode and transmit 360° images using existing video encoding technology, a projection must be made from the 360° data to a rectangular or sub-rectangular flat format. There are various standards for this format, many of them drawn from cartography rather than computer science, where they were originally chosen due to the relative qualities they possessed at expressing facts about a globe map. The trade-offs made are still relevant today, concerning issues such as the uniformity of distance between points and the relative amount of warping within the mapping.

Methods of rendering virtual environments with 360° views were examined by Ardouin et al.[46] who discussed various projections and their utility for a user interacting directly with a 360° view as their interface.

2.4.0.1 Mollweide

The Mollweide projection was proposed by Karl Brandon Mollweide in 1805. The projection trades accuracy of angle and shape for accuracy of proportions in area.

2.4.0.2 Hammer

Ernst Hammer in 1892 presented a variant projection inspired by the Mollweide projection, and enclosed within the same 2:1 ratio elliptical shape. Hammer's projection reduces distortion in regions of the outer meridians.

2.4.0.3 Conical

The conical projection, known more specifically as the Albers equal area projection, is a mapping formed by the projection of the globe first onto a cone and then the unwrapping of that cone, forming a mapping with two standard parallels at the top and bottom which 'cut off' unrepresentable peaks of the globe. Between these parallels however, distortion is minimal. This mapping is not suited to projection for computer graphics however, as it is missing data which the user may wish to see.

2.4.0.4 Equirectangular

The equirectangular projection, also known as the cylindrical projection, enjoys widespread use in photography and video.

2.4.0.5 Cube maps

A 360° can be expressed as a set of six cube faces, with each cube face taking the form of a perspective projection with a field of view of 90°. This is used within graphics as a means of representing 360° views within raster computer graphics. This forms the basis of the skybox technique [23]. The advantages of this format are that it is easily representable within 3D graphics. The disadvantages are that the cube now has ten discontinuity edges, and that distortion in the corners of the cube can be an issue if the image is saved in this format.

2.4.0.6 Tissot's indicatrix

The indicatrix proposed by Nicolas Auguste Tissot in 1859 for map distortion consists of overlaying on a globe uniform circular coloured areas, which are then projected onto the mapping. The relative warping on the circles then indicates which areas of the projection have received more or less warping. The circles represent distortion at a single point, but the distortion across the mapping is assumed to be continuous and smooth.

2.5 Summary

This chapter introduced several methods of storing caches such that successive rendered frames need only perform a subset of the rendering typically needed to draw a frame. This was considered both spatially and temporally, and as a combination of the two properties. A comprehensive review of this techniques both as they are considered here and as they can be applied to many more specific uses of rendering too many to describe at length in this thesis is available in Scherzer et al. [47]. In the next chapter, these methods will be leveraged as a means of transmitting less data when rendering a scene at one location and displaying it at another.

Chapter 3

Remote Rendering and Streaming Video

For short-sighted Men see remote Objects best in Old Age, and therefore they are accounted to have the most lasting Eyes.

— Isaac Newton, *Opticks*

The content in this chapter constitutes a review of techniques which will be relevant for Chapters 7 and Chapter 8 of this thesis. Specifically, related work in the areas of remote rendering and providing virtual reality experiences will be reviewed, as well as material concerning 360° mappings and projections. The critical assessment of this material will be performed in Chapter 5.

State-of-the-art methods of compressing and decompressing rendered scenes are introduced and described, both for decreasing render time and efficient transmission of data. This chapter will cover both traditional solutions to transmitting rendered graphics from one location to another as well as rendering-specific solutions, focusing on augmenting traditional video encoding.

3.1 Remote rendering

To achieve high quality graphics often dedicated hardware is used. Animation companies utilise banks of rendering machines to generate high quality works in progress to be reviewed, and

computer games have often depended on discrete graphics hardware to accelerate the graphics pipeline.

This hardware may not be particularly portable. Supercomputers are physically large, and it is may be a costly proposition to have to move this hardware to any given location where graphics may be required to be rendered. Nonetheless, a rendered scene may be required to be displayed in real-time at a remote location.

Efficient modern video codecs use motion vectors to predict the movement of parts of the image across the screen, allowing a rate of less than one bit per pixel to be achieved. Motion vectors can be calculated much more easily in a rendering engine than by analysis of video alone. In McMillan et al. the cylindrical projection used to generate imagery makes the production of a motion flow straightforward, as all motion in the scene becomes motion across the surface of the cylinder, unlike in a more standard perspective view where motion is necessarily warped[49].

3.1.1 Streaming video

The most immediately straightforward method of taking a rendered scene and transmitting the data is to consider the continuous viewport as a stream of video data and compress and transmit it using traditional video compression tools. The most prominent video codec currently is H.264/AVC, a standard with wide hardware compatibility for both encoding and decoding.

3.1.1.1 H.264/AVC

H.264/AVC is the successor to MPEG-4 Part 2 video and MPEG-2 video. It continues the joint effort by the ITU-T VCEG and ISO/IEC MPEG standards committees to publish video standards for both general use and broadcast. As it is targeted to broadcast, it has features which are of interest in the context of remote rendering as well as in efficient video compression.

The wide support for H.264 is manifested in the presence of dedicated H.264 encoding and decoding chips in consumer hardware. Any given consumer device with a screen is likely to have such a chip present for both capturing from an inbuilt camera and playing purchased video content without draining a battery or overtaxing a cheap CPU.

H.264/AVC, like the codecs which preceded it, has its origins in the Discrete Cosine Transformation (DCT). It is a YUV (luminance and two colour difference channels) format and is

generally chroma-subsampled, although the specification supports 4:4:4 formats [50].

Discrete cosine transform The Discrete cosine transform (DCT) is an approximation to a Fourier transformation, which takes a set of discrete data points and expresses them as a sum of distinct cosine waves [51]. It is utilised in lossy compression because quality can be controlled by raising or lowering the amount of high frequency terms stored.

For a data sequence $X(m), m = 0, 1, ..., M-1$, the DCT is defined as:

$$G_x(0) = \frac{\sqrt{2}}{M} \sum_{m=0}^{M} X(m) \tag{3.1}$$

$$G_x(k) = \frac{2}{M} \sum_{m=0}^{M} X(m) cos \frac{(2m+1)k\pi}{2M}, \quad k = 1, 2, ..., M-1 \tag{3.2}$$

Where $G_x(k)$ is the k-th DCT coefficient.

The type-II DCT is a central part of JPEG compression [52]. Each channel of an image is split into 8×8 macroblocks which are converted to frequency space with a two-dimensional DCT.

$$G_{u,v}(k) = \frac{1}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} X(m) cos \frac{(2x+1)u\pi}{16} cos \frac{(2y+1)v\pi}{16} \tag{3.3}$$

Where (u, v) are the indices of the pixel in question in the macroblock matrix. Thus transformed, the values in the matrix now express from top left to bottom right the low to high frequency elements of the 8×8 block. This matrix is divided by a fixed quality quantisation matrix defined in the specification and rounded to integers. These integers are then losslessly entropy encoded (see Section 3.1.1.6) and subsequently Huffman coded.

H.264 integer transform H.264 advances on earlier MPEG video codecs which followed JPEG in their use of the DCT, in that it uses an integer transformation which closely follows the DCT but which can be computed entirely in 16-bit integer space with no loss of quality, as opposed to the DCT which from its mathematical basis involves floating point operations and floating point rounding [50].

Chroma subsampling The Human Visual System (HVS) detects chrominance changes much less accurately than luminance changes. For this reason, video encoding systems have long looked first to cutting the chroma channels down in efforts to compress video. The three most common formats are referred to as 4:4:4, 4:2:2:, and 4:2:0.

4:4:4 An equal proportion of data is used to encode luminance and the two chrominances.

4:2:2 A full resolution luminance channel is paired with two vertical resolution halved chrominance channels.

4:2:0 A full resolution luminance channel is paired with two chrominance channels halved in both horizontal and vertical resolution, to produce two quarter-sized images.

Frame types H.264 supports three types of encoded picture, varying in their temporal dependencies.

Intra-frame or I-frame Intra frames have no temporal dependencies, i.e. the information contained in them is sufficient to reconstruct the image in its entirety. It is functionally equivalent to a format like JPEG, with the exception of being stored in a video stream.

H.264 is not dependant on I-frames. It supports a feature called periodic intra refresh, whereby slices of the image can be refreshed independently. This is useful for hardware streaming, where the intra frame data can be split over several frames so as to keep the size of the buffer needed to receive data fixed.

Forward predictive frame or P-frame A P-frame is an inter frame which contains motion vectors informing a transformation on a previous I-frame or (rarely) otherwise in order to display itself. These are distinct from B-frames which are yet to come because while they rely on motion compensation, they do not require the data stream to be interpreted out of order.

Bi-directionally predictive frame or B-frame A B-frame is an inter frame which contains motion vectors information a transformation on one or more frames both backwards and forwards temporally in order to display itself.

3.1.1.2 H.265/HEVC

H.265/HEVC is the next-generation codec codified by ISO/IEC MPEG and the ITU-T VCEG as the replacement for H.264 [53]. H.265 is distinguished from H.264 by an increase in the available size of encoding blocks from 16x16 to 64x64, and an increased capacity to vary the size of blocks within a frame to best adapt the level of detail needed to preserve different regions. Where one of the discrete 64x64 blocks is sufficient to preserve detail, an efficiency gain is made over H.264, which would have had to describe multiple 16x16 blocks to encode the same area. H.265 also standardises image data specifications for 8-bit, 10-bit, 12-bit and 16-bit video, as opposed to H.264 which supported 8-bit and a limited extension to 10-bit.

3.1.1.3 AV1

AV1 is a next-generation video codec developed by the Alliance for Open Media [54]. It extends Google's work on the VP9 codec to provide a patent-free video codec for general use. Similar to H.265, it is a continuation of frequency-based approaches to video encoding, and supports large block sizes (up to 128x128) and predicting motion vectors in a higher bit depth than the first encoding. AV1 is intended to support 10-bit and 12-bit video, maintaining parity with H.265.

3.1.1.4 Conventional video codecs for streaming graphics

Within the context of streaming graphics, encoding the graphics stream as video is attractive because it is intuitive: if graphics are to be delivered to a client, simply take the graphics, encode them and stream them using regular software and/or hardware and then decode them at the other end. Commercial solutions to streaming graphics such as NVIDIA Shield use exactly this approach, combined with an H.264 encoder onboard the graphics card with direct access to graphics memory.

There are however downsides to straightforward video encoding. If it is performed in hardware, limited control is available over the choices the encoding engine makes with the data given to it. Certain regions may be of particular importance, but the video encoder not knowing this may deprioritise them. If a software video encoder is used, performance is a huge issue - video encoding speed is CPU bound, as is CPU-based rendering.

In addition, sending a high enough bitrate video stream may be an inefficient use of network bandwidth. Given the amount of metadata available to the renderer which is not streamed alongside the video, for example high resolution depth information, there are potential reductions in capability at the client end.

H.264 is particularly adapted to video streaming because the bitstream, designed with an eye towards this application, is constructed from individually headered NAL units. NAL, or Network Abstraction Layer, is an organisation of the video data into chunks with a specified integer size. In a simplification, we can imagine each NAL representing one frame (of any variety; for streaming purposes the temporally attached frames are shifted into order of decoding, so a pattern IPBBPBBP becomes IPPPBBBB). As long as the NAL units are streamed and received in order, the video stream can be played.

3.1.1.5 MPEG-4 container format

A container format is a schema for packing data and metadata in one or more 'streams' into a single file. These are typically used to combine audio and video, for example in a file containing a movie or TV show. They are also used to combine music with information about the artist and/or album the music is from. The process of combining one or more streams in a container with time information to keep them in sync is called muxing. The container format can have a large impact on our ability to stream a file from one location to another over a network; if the reading a file is wholly dependant on the ability to seek backwards and forwards in the disk memory, then the whole file must be received by the client in order to begin to play the file.

The MPEG-4 container format (MPEG-4 Part 14 [55]) is the most widely used container format for H.264/AVC video. It was released alongside H.263 and supports encapsulating a wide range of data formats, from multiple video streams to arbitrary binary data in packets fixed with timecodes to be demuxed when appropriate. It can be streamed either by splitting the file into the contained packets and sending them in order with no more sophisticated control, or else it can be transmitted with a more controlled means such as the Real-time Transport Protocol (RTP).

3.1.1.6 Entropy coding

As mentioned before, JPEG and the codecs which follow it use entropy encoding as the final stage of their compression [52]. This encoding runs through the frequency-space macroblocks in a zig-zag pattern (to maximise spatial coherence; the frequency in the two dimensions decreases at the same rate within the block) and run-length encodes them. Run-length encoding codes sequences of the same integer as an integer and then the number of times it repeats. The output of this is then Huffman encoded [56]. Huffman encoding is an optimal prefix encoder which uses a frequency-sorted binary tree.

3.1.2 Rendering directly to a video stream

The close relation between the methods used by video encoders and the data turned over as a matter of course in rendering engines has long been noted. A large part of the efficiency of modern video codecs is down to the use of motion vectors to predict the movement of chunks of image across the screen, allowing rate of less than one bit per pixel to be achieved. Motion vectors can be calculated much more easily in a rendering engine than by analysis of video alone, as in [49] where the cylindrical projection used to generate imagery make the production of a motion flow trivial. Motion vectors in other systems may already be being produced, in the course of saliency calculations and similar.

3.1.2.1 Calculating motion vectors with a renderer

Wallach et al. described a method of passing motion vectors straight from a renderer to the then-state-of-the-art MPEG video encoder. They showed significant efficiency improvements over the use of exhaustive search methods [57].

3.1.2.2 Frequency-space rendering

More closely ingrained with video encoding still, Bolin et al. [58] described a ray tracing system which calculated its output already in frequency space, producing still images which were already in a pseudo-JPEG format. This allowed the renderer, a Monte-Carlo based ray tracer, to prioritise low-frequency areas where noise would be apparent with more rays and high-frequency areas where noise blends with fine detail less rays. Thus, the system has positive gains of both

improved efficiency of rendering and lacking a need to be run through an encoder to produce compressed output - the output is self-compressed.

Herzog et al. [59] advance this method to inputing straight to the MPEG-2 video encoder, combined with a lighting approach which favours temporal coherence for efficient motion flow generation. Their system also measures the error accorded in individual DCT blocks and uses it to guide the renderer according to perceptual thresholds.

3.1.2.3 Difference images

Levoy [60] proposed a system for mixing server and client rendering based on the ability of a server to render high quality textures and complex shading, while relying on a lower-powered client to render information such as smooth shading and sharp edges, on the basis that such information performs poorly in typical video encoders. This depended on calculating both a high-quality and low-quality rendering on the server end and then encoding and streaming with a video encoder the difference between the two, on the basis that the difference added to the low-quality render produced simultaneously on the client would mathematically produce the original high-quality render. Video codecs are not suited to encoding difference information however, and combined with their inherently lossy nature the output will not match the original image exactly.

3.1.3 Video augmented with rendering metadata

Drawing on the methods used for spatio-temporal upsampling, Pajak et al. [39] propose a system for remote streaming of graphics which lessens the demands of streaming high-resolution video by pairing a low-resolution video with efficient encoding of various metadata buffers, and producing the final image using the (otherwise unused) processing capabilities of the client. The cost of this approach is that more resources are required to be present at the client end. An approach based on video streaming alone might require the client only to possess a simple hardware-based video decoder, as commonly found in mobile phones and similar devices. An edge-encoding algorithm, unless a hardware solution for a specific implementation was produced, would require increased general purpose processing power, with the proposed benefit of improvement in image quality and a reduction in required bandwidth.

The streams of data required to be sent are a low-resolution rendering of the scene, combined with an edge encoding of the depth buffer and an edge encoding of the object motion in the scene. These are decoded at the client end, and the motion buffer encoded is combined with a client-end calculation of eye movement, and then the aforementioned upsampling is performed to produce the final image, albeit without high-resolution disocclusions to depend on. For this reason, the low-resolution image is still recommended to be larger than that recommended for the spatio-temporal upsampling (i.e. 2×2 blocks rather than 4×4).

3.1.3.1 Low-resolution video stream

The low-resolution video stream is generated by a jittered camera to ensure that a full scan of each high-resolution output frame is produced every Δn frames.

3.1.3.2 Metadata buffer encoding and streaming

Metadata buffers have unique challenges and opportunities for streaming data [44]. They are simpler to represent visually than full image data video in that they tend towards flat colours or smooth gradients with well-defined edges. This has the negative impact, though, that techniques in common video codecs aimed at accommodating the complicated data in images may be wasteful when processing the simplified data. For example, a high bit depth (e.g. 16-bit) one-dimensional depth buffer passed into a non-adapted H.264 encoder will have 8 bits of luminance information to be encoded in, and two channels of irrelevant chrominance data will be stored with the result. Specialised methods are developed both to adapt metadata buffers to standard video codecs, as well as developing wholly different encoding schemes for such data.

3.1.3.3 Edge detection

Edge detection is commonly performed with a Laplacian filter, which detects changes in the second derivative of a pixel function $L(x, y)$.

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{3.4}$$

It is commonly achieved using an image processing kernel, a figurative piece of code which operates per-pixel informed by the neighbourhood of the pixel it has reached. Figure 3.1 shows

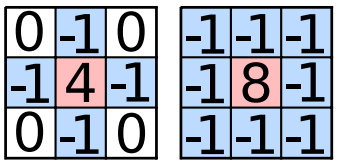


Figure 3.1 – A Laplacian kernel detects edges with a weighted box filter. In a smooth texture or a gradient, the centre value will be roughly equal to the sum of the surrounding values. At an pixel edge, there will be a larger difference.

two typical Laplacian kernels, one for horizontal and vertical edges, and one which will match diagonal edges also.

3.1.3.4 Streaming metadata buffers as video

Pece et al. proposed a custom schema for encoding a continuous stream of depth buffer information using regular video codecs [44]. Working on a 16-bit unsigned integer buffer, the depth information is packed into the luminance and two chrominance channels of a regular video stream (i.e. H.264). The luminance channel is a straightforward linear packing of the buffer.

$$L(d) = \frac{d + \frac{1}{2}}{2^{16}} \tag{3.5}$$

The chrominance channels are provided a piecewise linear triangle wave each, sufficiently fast-changing to be represented meaningfully in a potentially subsampled chroma stream.

$$H_a(d) = \begin{cases} \frac{L(d)}{\frac{p}{2}} \bmod 2 & \text{if } (\frac{L(d)}{\frac{p}{2}} \bmod 2) \leq 1 \\ 2 - (\frac{L(d)}{\frac{p}{2}} \bmod 2) & \text{otherwise} \end{cases} \tag{3.6}$$

$$H_b(d) = \begin{cases} \frac{L(d) - \frac{p}{4}}{\frac{p}{2}} \bmod 2 & \text{if } (\frac{L(d) - \frac{p}{4}}{\frac{p}{2}} \bmod 2) \leq 1 \\ 2 - (\frac{L(d) - \frac{p}{4}}{\frac{p}{2}} \bmod 2) & \text{otherwise} \end{cases} \tag{3.7}$$

Where p is chosen to be more than twice the ratio of the packed bit depth to the unpacked bit depth. This approach can easily be tabulated and as such is very computationally cheap to implement on top of an existing system, but depending on being able to encode a full-size video stream in addition to any other considerations in a rendering engine is cost enough.

3.1.3.5 Binary map encoding

As part of the remote rendering system detailed, [39] proposes a metadata encoding not dependant on traditional video streaming. The depth and motion buffers are both encoded using a laplacian operator to produce edges, which are then used as a map to provide edge samples. A regular sample of each 32×32 pixels is added to the edge samples in order to encode gradual changes.

The samples are then streamed as two sets of data. First, a binary image map indicating the location of the edge samples encoded with JBIG2, an effective scheme for lossless binary image encoding [61]. JBIG2 implements adaptive binary arithmetic coding [62], a coding which models probabilities based on the spatial neighbourhoods of the characters it receives. This allows it to achieve less than one bit per symbol on average, minimising the impact of separating the map from the data.

Secondly, the data is minimised to remove the unsampled regions using a predictor which informs the depth buffer based on the motion buffer, and the motion buffer and depth buffer based on a map of the sign of the laplacian operator, as due to the laplacian operator maintaining sign along edges, samples which are spatially close will also likely have the same sign. The predictor used for this purpose by Pajak et al. is Paeth Prediction [63] followed by entropy coding.

3.1.3.6 Depth and motion buffer decoding

The depth and motion buffer samples are decoded from their compressed binary format at the client. They are then extrapolated by an adapted push-pull algorithm [64] into high-resolution buffers which are used to inform the upsampling of the low-resolution video stream.

The push-pull algorithm functions by placing the edge samples into a full sized buffer and then repeatedly downsampling it until the sample information fills the entire downsampled buffer. These buffers are then upsampled, using the data from the topmost buffer up through the levels until the original edge samples are placed in. In this way the high-resolution edges remain crisp, while the samples within them are diffused. As the regular sample was 32×32, a factor of 2 on each edge results in a maximum of 5 downsampled buffers for any input.

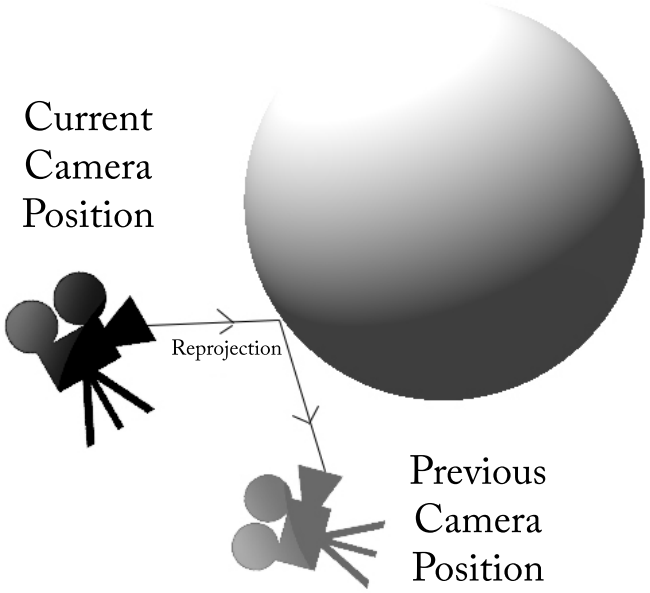


Figure 3.2 – Camera position reprojection for motion buffer generation.

3.1.3.7 Composition

To reconstruct the final image, the system proposed in Pajak et al. [39] relies on the reconstructed metadata buffers, the low-resolution rendered frame, and the available client-end previously reconstructed high-resolution previous frame buffer. Using the motion data, pixels in the current frame can be projected back into the previous one (shown in Figure 3.2), taking advantage of temporal caching methods as discussed earlier. This is mixed with the spatially upsampled low-resolution video stream using a bilateral weighting system. As there is no scope to request back to the server for disoccluded pixels, these must depend on the upsampled low-resolution buffer alone.

3.1.3.8 Performance

The method described is claimed to achieve performance similar to that of a high-end video encoder. It suffers, however, when asked to transmit effects such as reflection where the computed motion flow will not match the actual optical flow. This could potentially be improved on by transmitting further metadata buffers specifically for such optical effects, but at the cost

of degrading performance compared to traditional video encoding. In addition, where the temporal upsampling fails due to disocclusion, the image quality appears significantly reduced in still images, though the effect is mitigated in video.

3.1.4 Remotely transmitting pre-computed lighting information

Crassin et al. [65] published a review of methods of remote rendering pursued using cloud-based servers to pre-compute indirect lighting from three different algorithms. Pre-computing and then transmitting the lighting as a video stream by use of GPU-based hardware H.264 encoders permitted the engagement of both server and client-side graphics hardware, for multiple cloud servers rendering to multiple clients, amortising the cost of the original indirect lighting computation.

The three methods of indirect lighting used have distinct implementation differences.

Voxels

The implementation of indirect lighting used in the Voxels case is that of Crassin et al. [66]. The Voxels represent indirect irradiance as a directional quantity on a sparse 3D representation of a scene, relatively inexpensively. The voxels themselves are prohibitively large to transmit, so the method described computes a full image on the server which is encoded with H.264 and transmitted to the client, which then adds direct lighting to the scene using local graphics hardware.

Irradiance maps

The Irradiance Maps used in this method are described in Mitchell et al. [67]. The light maps are textures, transmitted to potentially multiple users indiscriminately. Again local GPU hardware is required in order to add direct lighting to the objects the light maps belong to, but in this case the server end need not render a full resolution image specific to the client.

Photon mapping

Real-time photon mapping was described in Mara et al.[68]. This approach is the most technologically flexible described, as independent photons could be parallelised over multiple GPUs in the cloud. It is however a computationally expensive procedure at the client end as

the photons themselves are transmitted losslessly to the client, which must perform the rest of the procedure as well as coordinate. This cost limits the deployment of this method to high-end workstations.

While the methods described form a survey of indirect lighting methods for cloud computing, none of them individually fulfils the full scope of the remote rendering problem - Voxels and Irradiance Maps depend on a video codec ill adapted to their particular purpose, while Photon Mapping retains high client hardware requirements.

3.1.5 Streaming stereo imagery

A common way of representing stereo imagery in video and in static images, going back to Victorian stereoscopes, is to present the left and right image either side-by-side or top-and-bottom and, in processing them, cut the image in half and present half to either eye. This is considered an inefficient method of storage as it does not take advantage of the duplication of imagery between the two eyes. In addition, encoding artefacts can occur on the edge where the two images are joined.

Warping has been proposed as a solution in graphics for the generation of a second eye perspective for stereo [39]. As most of the content in the image is shared between the left and right eyes, one image may be warped to provide the majority of the other and then by filling in any disocclusions and removing any occlusions an accurate image can be formed with much less draw on resources than a full second render.Didyk et al.[69] described a method of adaptively scaling block sizes for reprojection based on disparity to provide a stereo view for a perspective camera in a virtual environment. The method proposes using a push-pull algorithm for filling gaps.

3.2 Remote rendering for virtual reality

Current commercial approaches to streaming graphics to HMDs tackle the problem by sending a single H.264 stream of the final processed graphic for the HMD, including warping for the HMD lenses, side-by-side. This can introduce unique issues into the display pipeline, such as encoding artefacts corrupting the per-pixel processing to adapt to the HMD lenses.

The issue of latency comes to the fore when considering remote rendering for virtual reality.

When rendering a live viewpoint to a HMD, time is already constrained due to the requirements of users that the image update at a high frame to appear realistic, but now also the latency between the user and response must be correspondingly high for the remote server to update the image in time in response to the users movement.

3.3 Summary

The techniques presented in this chapter concerning remote rendering and the augmentation of a stream of rendered graphics provide the background for chapter 7 and chapter 8, in which research is presented building on these concepts to propose novel remote rendering methods.

There are distinctly different approaches to remote rendering. Video streaming takes most immediate advantage of much existing infrastructure, but is a blunt tool which does not take advantage of the unique placement of computer graphics as being fully expressed in memory at each frame. Alternative, more sophisticated methods are being developed, which provide advantages to the client in many ways. Remote rendering is essential to providing high quality rendering in locations where it would be impractical to take large computing equipment.

Chapter 4

High Dynamic Range Imaging

> *Whether a dress, a house, or a flower is beautiful is a matter upon which one declines to allow one's judgment to be swayed by any reasons or principles. We want to get a look at the object with our own eyes, just as if our delight depended on sensation.*
>
> — Immanuel Kant, *Critique of Judgement*

This chapter details the relevant background information on HDR imaging, including the concepts and methods used in encoding High Dynamic Range video in real time. Chapter 6 concerns the end-to-end HDR pipeline, and Chapter 7 and Chapter 8 use these concepts in the context of remote rendering.

4.1 High dynamic range imaging

High dynamic range imaging (HDRi, or HDR) allows the full range of lighting in a real world scene to be captured, stored, transmitted and displayed. Traditional computer imaging can be classified as low dynamic range, as can the monitors used to display it and the cameras used to capture real world scenes in representation.

4.1.0.1 Dynamic range

Dynamic range in imaging is the ratio between the brightest and darkest regions of detail captured within a single image. The Human Visual System can perceive roughly 14.5 stops at once [5], but can perceive greater ranges through physical adaptation of the eye.

Dynamic range can be measured in the logarithmic unit stops, which represent doublings of light intensity.

4.1.0.2 Standard imaging

Standard computer imaging systems for both still images (i.e. JPEG) or video (i.e. H.264) use 8 bits of information to store the luminance and each of the chrominance channels of their data. In practice, this means that they have trouble storing more than 8 stops (powers of two) of dynamic range. Curves applied to the data may help, but there is a fundamental limit and it is a low one.

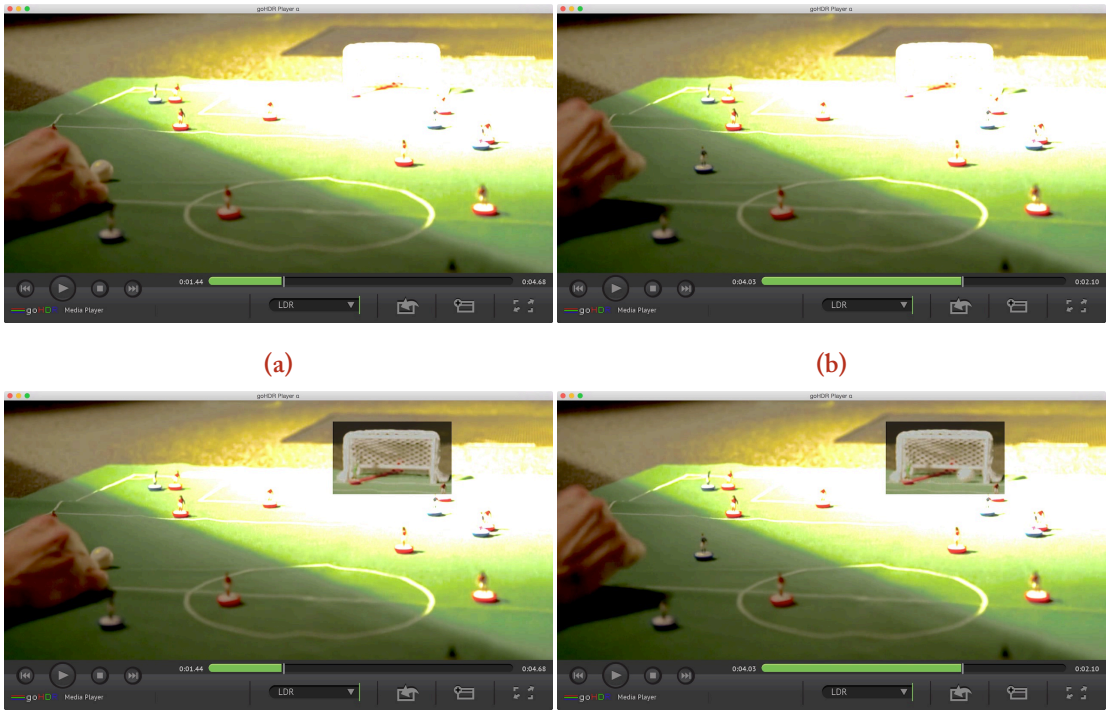
This range of data is not suitable for rendering, where typically reflections and refractions need to be measured which may be inherently scoped in range outside of what you can fit in $0-255$ values. Thus, graphics rendering generally stores light values as floating point numbers [23]. Floating point numbers allow for the storage of values from negative infinity through to positive infinity in a well-formed mathematical environment. As light is perceived logarithmically and floating point precision drops off with magnitude, the two are well paired.

4.1.0.3 Display

HDR imaging can typically be displayed on low dynamic range displays by clamping the range of values represented, above and below which appear as black (underexposed) or white (overexposed) as defined by the display in question.

The company Brightside pioneered the first HDR displays based on two technologies: a digital light processing (DLP) projector, and light emitting diodes (LEDs) [71]. The brightness of the Brightside DR37-P prototype HDR display was subsequently characterised using a spectroradiometer to record spectral radiance, chromaticities and luminance [72]. From this data the true increase in gamut of the display due to the additional LED layer was estimated. The colour space of the HDR displays have similar chromaticities to any other 8-bit per channel device, so there is no increased chromaticity resolution, but it is in the luminance dimension where the colour space is perceived to increase resulting in 35,961,042 potential colours compared to the 16,777,216 of a typical LCD display.

In early 2015, Samsung announced their TV model UN65JS9500 that can reach a peak



(a) (b)

(c) (d)

Figure 4.1 – (a) Ball is visible as it is struck, but (b) Goal cannot be seen due to over-exposed area (c) Pixels around goal modulated to allow visibility (d) Goal!

luminance of 1,000 cd/m^2 [73], however this is still some way in terms of dynamic range from the non-consumer HDR displays developed SIM2 which are currently capable of up to 6,000 cd/m^2. SIM2's monitor comprises an HD resolution (1920 × 1080) LCD panel. The high brightness is obtained by an array of powerful white LEDs which replace the backlight of the LCD panel [74].

Figure 4.1 demonstrates information which is out-of-range for display on a standard display being clamped, and how that information can be recovered from the HDR data by local adaptation.

4.1.0.4 Tone mapping

In order to adapt a higher dynamic range image to a display which is not capable of displaying a full image, a function known as a tone mapper can be applied to the image. There are many tone mappers available, each with their own strengths and weaknesses for different purposes.

An important distinction is between global and local tone mappers.

Global

Global tone mappers apply a function such as a sigmoid curve universally across all values in the image. This has the advantage of being fast and easily parallelised, but global tone mappers tend to be more restricted in the aesthetic quality of the image they produce due to low levels of relative contrast between objects.

Local

Local tone mappers regard the area around a given pixel in an image in order to best utilise the available dynamic range. This has the effect of maintaining relative contrast between regions, the cost of causing the image to appear flat or otherwise unworldly.

4.2 High dynamic range video

While for still imagery high dynamic range imaging has entered the mainstream and comes as a feature on many commercial products [75], high dynamic range video has yet to achieve such popularity. Nonetheless, many methods of encoding such video have been published and/or patented [4]. For the purpose of remote rendering, high dynamic range video is interesting as the imagery which comes out of the rendering engine is necessarily high in dynamic range, and is almost always taken and clamped to the range displayable on low dynamic range screens or storable and transmittable in low dynamic range video files.

With a high dynamic range video format, it is possible for the client to receive more lighting information than it needs to display in order to make judgements at the client end regarding display of the data. For traditional video files, this can mean adapting the display of the data automatically to best present it in the lighting conditions of the viewer. For rendering, it may allow the client to exercise more judgement in its post-processing of the rendered scene.

4.2.1 Classification

There are two generic approaches to encoding HDR video with different capabilities. Firstly, existing video methods can have their mathematical basis extended so that they take in higher

fixed point bit depths, i.e. 10-bit, 12-bit or 16-bit integer inputs. Combined with gamma curves, these codecs are able to take in a higher dynamic range than traditional video codecs. These encoders do not, however, take in a full floating point range. As well, there are few implementations beyond 10-bit and the implementations there are do not operate in real-time.

The other existing method by which HDR video is currently encoded is by processing a stream of tone mapped low dynamic range video with a video encoder such as H.264, and then accompanying it with a second stream enhancement layer, typically containing some residual data which permits the reconstruction of the original image. The advantage of this method is that it can allow the use of existing video encoding hardware and software, with only the reconstruction of the two streams needed to be additionally processed.

4.2.2 Inverse tone mapping

Potentially useful to this second method is the notion of inverse tone mapping as described by Banterle et al. [76], calculating the inverse of a tone mapping function and applying that to an encoded image to produce an image with the original dynamic range. Such reconstructed images, while they preserve range, introduce banding where the 255 values available in 8 bits were not able to quantise gradients effectively.

Given such a reconstructed image, the extra stream which is encoded can be used to compensate for the banding.

4.2.3 Existing formats

Mantiuk et al. described a system for encoding HDR video which followed the first course, in that it allocated an 11-bit luminance bitstream to go with the regular 8-bit chroma bitstreams (pre-chroma subsampling) [77]. A mapping function is provided to take a full range of real-world luminance information into the 11-bit space. The method however encounters noise artefacts caused by the discrete cosine transform interacting poorly with high in local dynamic range blocks, and so a second bitstream is also present as well as a higher bitrate video stream, in order to perform block edge corrections. This method is weakened further by the lack of wide support for 11-bit H.264 encoders.

A commercial method was proposed by goHDR [78] in 2011. It operates according to

the second generic model, whereby it extracts a base frame of tone mapped luminance from a high dynamic range image and uses it to produce by bilateral filtering an image containing the edge data and chroma information. These two streams are then encoded with a regular H.264 encoder. As there are two video streams, the output video is larger than a regular low dynamic range video, and the method is not backwards compatible.

HDR imaging allows the full range of lighting in a real world scene to be captured, stored, transmitted and displayed. Dynamic range, in imaging, is defined as the ratio between the brightest and darkest regions of detail captured within a single image. It is typically measured in logarithmic units *stops*, which represent doublings of light intensity. Traditional imaging systems, so called Low Dynamic Range (LDR), also known as Standard Dynamic Range (SDR), can capture and display about 8 stops. The Human Visual System through eye adaption can perceive detail in scenes ranging from starlight to bright sunshine, and in a single scene about 20 stops with minimum adaption.

While LDR imaging systems use 8 bits of information to store the luminance and each of the chrominance channels, HDR requires floating point numbers to represent the full range of light in a scene. With 32 bits required per colour channel, i.e. 96 bits per pixel, a single High Definition (1920×1080) HDR frame requires 24 Mbytes. At a frame rate of 30 fps, a minute of HDR video results in 42 GBytes of data, compared to just 9 GBytes for a minute of LDR data. Efficient data formats and compression techniques are thus essential in order to cope with the large data requirements of HDR data on existing infrastructures. Several pixel formats have been proposed to represent HDR pixel values, including shared exponent RGBE encoding used in the Radiance image format [79] and LogLuv encoding used in a custom HDR version of the TIFF images [80]. They are not, however, sufficiently efficient to be adopted as a standard for HDR pixel representation. Currently, the most widely used file format for individual HDR frames is OpenEXR.

Developed for the movie industry, OpenEXR has been released as free software under a modified BSD license, with the most recent release appearing in August 2014 [81]. OpenEXR is not, however, suitable for the representation of HDR video as sequences must be stored as discrete frames - without any form of inter-frame compression.

A number of consumer camera systems already claim to provide HDR video, such as the

mobile phone cameras in the Sony Xperia series and the Samsung Galaxy S6. While these cameras may capture a wider dynamic range, the data is only available as tone mapped LDR footage. Tone mapping is a means of attempting to preserve the perceived contrast in a scene within the constraints of an LDR infrastructure. A wide variety of tone mapping operators (TMOs) have been proposed (see Banterle et al. (2011) [4]). The majority of these have been for static images, but more recently a number of TMOs for HDR video have been proposed which take into account potential temporal artefacts [82, 83]. When it comes to displays, most are currently LDR, although a commercial HDR display, the SIM2 HDR47ES6MB, capable of displaying a peak luminance of 6,000 cd/m^2 is available [74]. Where the dynamic range of the display matches that of the delivered lighting, the images can be displayed in a straightforward manner [71]. If this is not the case then the images can be simply clipped to the lower dynamic range, or preferably tone mapped to provide an enhanced viewing experience on the traditional display.

4.2.4 End-to-end HDR pipeline

Users are continuously demanding improved realism and quality of captured and displayed images. HDR imaging captures and delivers the actual light present in a real world scene. By faithfully representing the wide range of intensities that are present in the scene (indeed, potentially more than the human eye can see), HDR imaging is *scene-referred* rather than *display-referred*, encompassing in theory all future imaging systems intended for the human eye.

4.2.5 Capture

Well known techniques of HDR imaging use multiple exposures with different exposure times to create a static HDR image [84]. This static approach fails when objects move causing ghosting artefacts. To minimise artefacts and still capture a large dynamic range, dedicated HDR video systems use, for example, multiple sensors with the same integration time through a single lens [85, 86].

The Kennedy Space Center (KSC), together with goHDR and the University of Warwick, has recently undertaken a detailed evaluation of the dynamic range of 18 camera systems (including different modes of the same camera) they have been using to record rocket launches,

including the ARRI Alexa in two modes and a Canon 5D system derived from the work discussed in Chapter 6 [87].

4.2.6 Manipulation

Manipulation includes applications such as colour grading, post production, image-based lighting and scene analysis. We can distinguish between off-line processes for cinematographic or pre-recorded TV content and real-time processes for live transmissions. Off-line processes are done in general by using commercial editing tools such as DaVinci Resolve from BlackMagic, The Foundry's Nuke, etc. For real-time HDR video manipulation, there are GPU based solutions that offer half-float, single-float or 16-bit integer data manipulation resources. OpenCV includes GPU functions while Halide [88] provides a novel approach as a functional language that targets different compilers including CUDA and OpenCL. VicomTech showcased its first prototype for live HDR video manipulation, named Tebas, in the Futures Park section of the NAB 2014 exhibition. In this system, the captured material was captured and pre-processed by goHDR's Flare HDR video system. It was then processed and mixed by Tebas before finally being displayed on a SIM2 monitor [15].

4.2.7 Encoding, storage and transmission

As discussed above, a key problem with HDR video systems is the amount of data that is generated. Efficient data formats and compression techniques are thus essential in order to cope with the large data requirements of HDR video data on existing infrastructures. Reducing the volume of digital data benefits areas including, but not limited to: a reduction of transmission channel bandwidth; a decrease of the buffering and storage requirement; and a reduction of data-transmission time at a given rate.

HDR video compression may be classified as one-stream or two-stream [13]. The one-stream approach utilises a single layer transfer function to map the HDR content to a fixed number of bits, typically 10 to 12 [77, 89, 90, 91, 92]. While it is true that many scenes do contain a dynamic range of lighting that is sufficiently low to be adequately contained within a limited number of bits, there are many others for which 10 or even 12 bits may be insufficient [13].

In a two-stream method the encoder has one input HDR video and two bit streams as an output [93, 94, 95, 96]. These streams can consist of (1) a standard compliant bit stream, for example HEVC Main 10, H.264 etc. and (2) one another stream corresponding to additional data to reconstruct HDR video (which can also consist of a standard bit stream) or metadata (if so desired). When combined these streams reproduce the full HDR content with minimal (or indeed no) perceptual loss. A process flow diagram for a generalised two-stream method is shown in Figure 4.2.

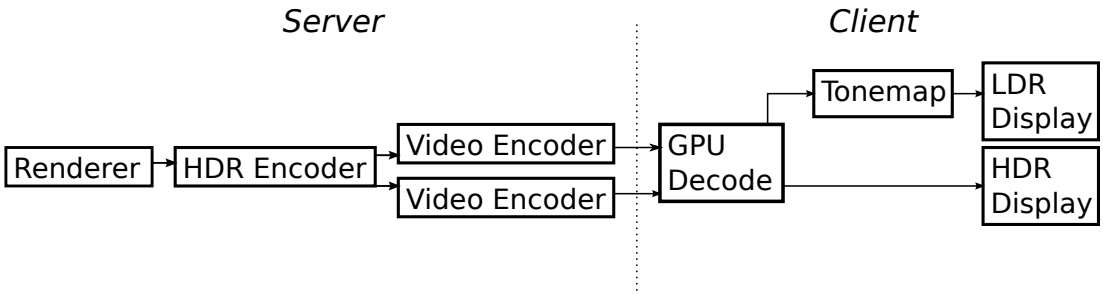


Figure 4.2 – Flow chart for two-stream HDR transmission method.

One-stream HDR video compression methods take advantage of the higher bit depth encodings possible using of modern encoding standards, typically 10-bit HEVC [97], to transform a single HDR video input stream into a single compressed stream using a transfer function [13]. In SMPTE ST2084 [98] this transfer function is the PQ_curve [91], while ARIB STD-B67 [99] uses Hybrid Log Gamma (HLG) [92]. A Power Transfer Function (PTF) is a one-stream HDR video compression method that uses a power function, similar to the well-known Gamma function used in traditional, Low Dynamic Range (LDR) video. This has been shown recently to produce high quality HDR video compression very efficiently for a value of $\gamma = 4$ [100]. In Chapter 7 and Chapter 8 ST2084 is used as the method of HDR video compression.

4.3 Summary

HDR video, which is becoming commonplace in consumer electronics, represents one of the high-fidelity imaging frontiers which is discussed further in Chapter 6, Chapter 7 and Chapter 8. Chapter 6 discusses the implementation of a platform for streaming HDR video, which is utilised in the subsequent chapters.

Chapter 5

Research Focus & Methodology

> *It would, without doubt, seem odd to a mathematician to go about to convince him the diagrams he saw upon paper were not the figures, or even the likeness of the figures… in this science the reasonings are free from those inconveniences which attend the use of arbitrary signs, the very ideas themselves being copied out and exposed to view upon paper.*
>
> — George Berkeley, *An Essay Towards a New Theory of Vision*

The forthcoming chapter contains a discussion and critical assessment of the techniques for the capture, transmission, storage and display of imagery described thus far, and as concerns rendered computer graphics. A specific focus will be placed on the ability to incorporate new and emerging imaging technologies, as in Figure 5.1, which shows the different technologies applicable to each stage of the video capture-to-display pipeline. In light of the overview given of the area, a research question, methodology and objectives will be presented.

5.1 Assessing existing approaches to future imaging technologies

The common thread amongst the technologies and techniques for image display discussed in the previous chapters is that the nature of the stored and transmitted image is changing across multiple dimensions. The future display technologies, incorporating both traditional viewing screens as well as head-mounted displays and other virtual reality methods, are constantly broadening in approach and capabilities. HDR screens are now coming to market commercially, of-

fering screen brightnesses of up to 1000 nits [73], while for head-mounted displays pixel density is rapidly increasing, approaching or surpassing the display resolutions of mobile phone screens or broadcast TV streams. Figure 5.2 shows a matrix of imaging technologies and their intersection with the requirements of HMDs. Other sectors such as 360° video capture and storage have gained renewed interest and research through proximity to VR. All these concepts and more must be accommodated for in capture, processing, storage and display. Streaming, the focus of this thesis, amalgamates aspects of each of these stages, in that that the content to be streamed must be captured and processed in real time, then stored in the form of data packets to be transmitted before display.

In the following sections there will be a brief analysis of the state of the art in these areas, each one a representative area of high-fidelity imaging, to allow reflection on where there is work yet to be covered. With this presented, a roadmap will be assembled for the chapters of this thesis to be placed in context with their underlying motivation.

5.1.1 High dynamic range video processing

The field of HDR video research has greatly advanced in recent years with the advent of commercially available 10- and 12- bit video encoders, such as H.265 [102] and VP9 [103], and the standardisation of methods of packing HDR video for use with these. Where previously efforts in the compression of HDR video were restricted by the use of inefficient reference software for encoding and decoding, available software and hardware has now been made available which, through freely available high bit-depth encoders such as x265 [104], obsolete more impractical

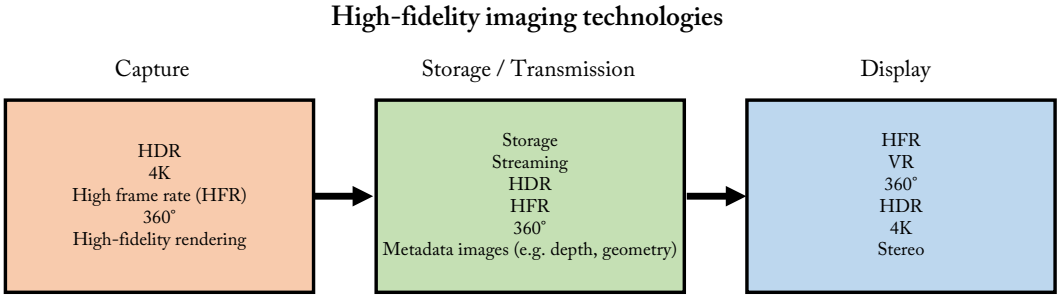


Figure 5.1 – The range of incoming technologies to accommodate with new imaging techniques, at every stage of the video pipeline.

	Present	Future
Standard displays	4K HDR (10-bit) HFR Wide Colour Gamut (10-bit)	12-bit+ HDR 12-bit+ Wide Colour Gamut Real-time high fidelity graphics
Head-mounted displays	360° Stereo	4K HFR HDR

Figure 5.2 – Some current and future technologies available in standard displays will present opportunities for HMD research in the near future. Meanwhile, some technical challenges are unique or specific, e.g. the enhanced importance of stereo imagery for HMDs.

methods and enable the efficient use of simple curves for encoding the full range of data, if possible. These standardised methods include SMPTE ST2084 [98] and ARIB STD-B67 [99].

Parallel to the development of these encoding methods, advanced generic libraries for the use of GPGPU work have become widely supported. OpenCL in particular is available across a range of both desktop and mobile devices. GPGPU presents a particular opportunity in the area of HDR video processing because GPU hardware is well adapted to handling full-range or half-range floating point images, required to correctly handle HDR data.

To match these efficient new techniques for video encoding it is therefore useful to conceive of a flexible new platform within which the encoding methods can be implemented, evaluated, tested and applied entirely within a GPU. To this end, there is an opportunity to develop a platform for processing full-range floating point images on a GPU, as a means of implementing various HDR video encoding methods.

5.1.2 Remote rendering

Existing methods of remote rendering, especially commercial solutions, focus on either real-time raster engine performance, such as in commercial remote gaming systems, or else in high-

latency systems for high-fidelity graphics. The issue is that high-fidelity graphics, when attempted in near real-time, produce high-noise images which encode poorly. This poor encoding quality is (of course) counter-productive to the intended high fidelity. There is therefore scope for a streaming system specific to high fidelity graphics which accommodates these particular technical challenges.

5.1.2.1 Streaming metadata buffers alongside rendered images

When the aim is to produce multiple viewpoints, or otherwise manipulate a high-fidelity image with information specific to the rendered scene, metadata sent alongside the images can be extremely useful. The typical example, as described by Herzog et al. [38], is streaming depth information to facilitate the use of a history buffer, but it is also possible to stream surface normals, an albedo map, a set of motion vectors, and any other per-pixel metadata capable of being produced.

5.1.2.2 Efficient encoding of information rendered using ray tracing

The question of lossy encoding wasting information which significant computing time was required to generate is a difficult one. To some degree, the two goals are directly contradictory. The ideal solution to this is to have the renderer aware of the video encoder's parameters and capable of avoiding areas likely to be compressed or capable of fitting the render directly to an output amenable to the encoder. This is extremely difficult though, and requires heavy integration between a renderer and an efficient video encoder in a way which prevents using standard video encoders, into which significant work and research have been made. A significant work towards this idea is Render2MPEG [59], but the utilisation of the MPEG-2 codec in that paper has lost pace with the uptake of the H.264 and H.265 codecs for internet streaming.

5.1.2.3 High dynamic range streaming and graphics

Future-orientated remote rendering must now consider the issue of remote rendering of HDR content. It is possible that, with knowledge of the curve used for encoding the HDR information, further accommodations could be to fit the HDR content coming out of the render into encoded video. For example, regions of the image which turn out to be above or below the dy-

namic range capable of being encoded could be de-prioritised, or the level of noise acceptable in dark regions could be dynamically adjusted according to the current display parameters [105].

5.1.3 Low latency streaming for virtual reality

The question of low latency, already important in remote rendering to provide a useful experience to the end user, is pushed to the forefront in remote rendering for virtual reality displays such as HMDs. Where a variable level of frame rate as low as 15 may be considered acceptable in a regular remote rendered display, a HMD requires a smooth image at as much as 75 frames per second to avoid inducing motion sickness in the user [9]. To this end, it is imperative that the client software be able to respond to head movement even outside of the response of the server - and indeed popular commercial offerings such as the Oculus Rift detect when the software being used has not displayed a new frame in time and update the display with simulated head movement.

5.1.3.1 360° video streaming for stereo virtual reality

360° video, which has achieved success in parallel to virtual reality HMDs, presents an attractive compromise for streaming images to them. As a video format which contains a full sphere of captured image data, 360° video can from a software counterpart to HMD technology. As every image possesses a full view, by mapping the stored video to a sphere any head movements which only change the view can be accommodated without the production of a new image, so long as there is no internal movement within the scene. Rendered graphics for HMDs using raster engines typically draws the projected geometry straight into the HMD viewport, but can be adapted to draw a full 360° view in various projections [46]. The downside to this approach is that either a larger, more detailed image must be rendered (and as such at a slower rate) than a single perspective, or else the quality will be hit as the majority of information in the generated image is no long presented on screen. In addition, a projection must be used to map the spherical image to the typically square or rectangular rendered video, disconnection the idea of the pixel from that of a square region.

5.1.3.2 Stereo streaming for virtual reality

Stereo, despite several popular resurgences, has never been a fixture of desktop consumer displays. With HMDs, however, it is essential to provide a realistic experience even on a level as basic as allowing the user to perceive depth when operating motion sensible controls. Didyk et al. [69] proposed a system for generating a stereo pair based on reprojection, but for single perspective viewpoints. Stereo imagery greatly complicates the use of 360° video, as the generation of a 360° image presupposes a single point of view onto the scene. 360° video players generally offer the user two fixed viewpoints within a single projected sphere to at least maintain an expanded field of view within the 360° environment, but depth perception is lost and objects within the scene will be perceived as stretched or warped where they ought to dis-occlude.

5.1.3.3 Server/client cooperation in virtual reality

As discussed above, the operation of a HMD is a resource intensive process. With this in mind, it is useful to consider the category of clients operating a remote rendering system which may have well performing graphics hardware, even if real-time high fidelity graphics are not possible. A system is conceivable in which a reasonable quality image is rendered in real-time on the client, then overlaid from the server end with a high-fidelity rendering which is constantly updated while in view.

It is an unnecessary cost in resources at both the server and client devices to have underused capacity for computation on client device. It is the purpose of intelligent compensation and prediction methods to strike the most efficient possible balance between the two, which will necessarily vary between classes of client (e.g. smartphones, workstations) and also between clients of differing age and capability.

5.2 Research question

Summarised into a single research question, which this thesis will attempt to answer, we can consider the problem at hand as follows:

- How can we adapt our methods for capture, transmission, storage, presentation and most

importantly streaming of image data to the challenges presented by incoming technologies for more detailed imaging, both on existing displays as well as in virtual reality?

5.3 Research methodology

In light of the areas highlighted in this section, this research attempts to make foray into filling the gaps across a number of future display problems. As the research develops organically, tools produced for one purpose are adapted to solve problems in other areas. This structure is shown in Figure 5.3. The framework constructed initially to aid in the real-time capture and transmission of HDR video is adapted first to produce metric results for the evaluation of that system for HDR, then to both enable and measure the performance of the subsequent proposed methods for remote rendering and virtual reality.

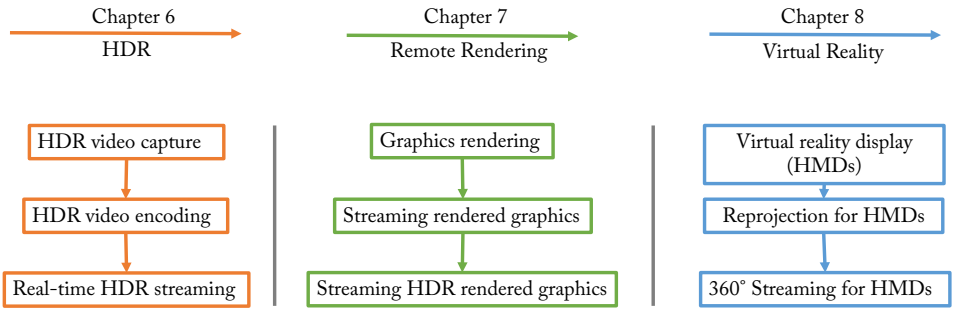


Figure 5.3 – The work in the thesis is structured into three primary branches. The first branch concerns HDR display, encoding and transmission. From here the work progresses to looking at hybrid remote rendering systems incorporating HDR encoding, and then the final section of work concerns virtual reality and HMDs.

This framework is developed as an approach to accelerating HDR video encoding using methods of GPGPU. Specifically, the OpenCL platform is chosen for its ability to measure GPU performance directly against CPU performance using the same code-base. Once video encoding is complete, a system is constructed to capture HDR video in real time and transmit it using first two cameras in a specialised rig, and then a single camera which is configurable

per-frame.

As almost all rendered imagery is naturally HDR in origin, rendered images present a special and significant case for HDR encoding and transmission. With this in mind, the research moves into looking at adapting existing methods of remote rendering, utilising depth and geometry buffers, to HDR encoding and HDR display. This system is evaluated on a basis of bandwidth, quality and performance against both its LDR predecessors and other intuitive methods of streaming HDR rendered imagery.

Subsequently, this technique will be expanded both at the server end, adapting it to generate and transmit 360° video, and at the client end, adapting the client to present the rendered image projected again to form a stereo view on a HMD. This thesis will propose a novel system using a free view client generated depth image to allow full stereo view 360° using reprojection. This system will be evaluated using objective image quality metrics for both LDR and HDR methods, broken down by component to present an image of the quality loss in each stage. The computing performance will be analysed on a standard workstation computer and the total performance will be assessed using a pre-rendered 'real-time' set of images to simulate having a real-time rendering supercomputer.

5.4 Research objectives

The objectives of this thesis are:

- To develop a framework through which HDR video can be analysed and processed in real-time.
- To develop a novel adaptation of existing remote rendering techniques to HDR video.
- To develop a novel approach to streaming high-fidelity HDR graphics to a HMD through reprojecting 360° spherical video.

5.5 Summary

In this chapter, techniques for the capture, transmission, storage and display of imagery, including HDR content, have been critically assessed. In light of this critique, the following

chapters will present the research performed in the course of this research, which explores these opportunities for further discovery.

Chapter 6

HDR Video Streaming

Thus, when the rays of the sun falling on the surface of water are reflected into our eyes, we imagine the sun as if it were in the water, though we are aware of its real position.

— Baruch de Spinoza, *Ethics*

This chapter describes an end-to-end HDR pipeline for capturing, encoding and streaming high-definition HDR video in real-time using off-the-shelf components. All the lighting that is captured by HDR-enabled consumer cameras is delivered via the pipeline to any display, including HDR displays and even mobile devices with minimum latency. The system thus provides an integrated HDR video pipeline that includes everything from capture to post-production, archival and storage, compression, transmission, and display. This chapter also presents the development of a GPGPU framework for the management and manipulation of HDR video, incorporating capture, transmission, encoding and display. This will form the cornerstone of the work in the subsequent chapters, as a platform more generally for performing arbitrary operations on per-frame imagery.

6.1 Commodity HDR video

There has not yet been an integrated pipeline that includes everything from capture to processing, storage, compression transmission and display in HDR [107].

In an end-to-end HDR pipeline, the full range of light in a scene, captured by an HDR-

enabled consumer camera, is delivered across the pipeline where it can be directly displayed on an HDR display, such as the SIM2. If there is no HDR screen available, the image can be tone mapped for display on a traditional LDR monitor, including mobile device displays. In addition, because the full range of light captured is preserved for every pixel along the entire pipeline to the display (so called true HDR), it is possible to modulate the exposure of each individual pixel, either automatically or by the user, to show detail in the scene that might otherwise not be visible. The viewer can adopt an active attitude, engaging with the video to dynamically alter the exposure of the entire scene, or a window within the scene. For example, as Figure 4.1 shows, the user is able to choose the exposure around the goal area to be able to see the goal as it scored, even when this is in bright "sunshine".

While partial HDR pipelines have previously been described [108], and shown at events such as IBC 2011 and IBC 2013 by goHDR and also subsequently at NAB 2015 by companies such as Dolby, the goal of the system presented in this thesis is to provide HDR video on commodity devices at HD resolution and at a frame-rate comparable with that of commercial LDR systems. In addition, this HDR pipeline is the first to incorporate real-time compression and streaming.

To ensure the wide embrace of HDR video, it has to be affordable and thus available for "every-day" use.

6.1.1 HDR-enabled consumer cameras

Magic Lantern (ML) is free software that improves the functionality of a range of Canon DSLR cameras [109]. In particular, ML provides a firmware patch which was capable of alternating the ISO exposure setting of a Canon 5D Mark III on a frame-by-frame basis. We adapted this firmware to perform the alternating exposures in real-time over the camera's HDMI interface. A Blackmagic capture card was used to input the stream of alternating exposures into a computer where a GPU based HDR merging algorithm combined the alternating exposures into single HDR frames. Deghosting methods, such as Ramírez Orozco et al. and Granados et al. (2013) [110, 111], are necessary to minimise any ghosting artefacts. As the results from the Kennedy Space Center shown in Figure 6.1 demonstrate, a single Canon 5D Mark III under this technique is capable of achieving 14 stops. While this is not true HDR, which is defined

by EU COST Action IC1005 [112] and MPEG [113] as > 16 stops, it is still a considerably wider range than the ~8 stops from off-the-shelf LDR cameras.

Table 6.1 – Evaluation of HDR video systems.

Camera - Mode	Stops, 0.5 RMS Noise	Native ISO	Vendor Spec.
ARRI Alexa - Log C	13.9	800	14.0
ARRI Alexa - Rec. 709	11.8	800	N/A
Canon 5D Mk. 3 - (ISO 400/1600)	11.4	100	8.0

6.1.1.1 Case study: Capturing a rocket launch in HDR

In 2013 the Advanced Imaging Lab (AIL) of the Kennedy Space Center was tasked with exploring current and emerging imaging technologies to help improve capture and display of all aspects of the US space program. In particular AIL wanted to evaluate HDR video performance while concurrently imaging the bright engine flame and the shadow areas of an Atlas 5 rocket launch, which current imaging technology is incapable of doing resulting in substantial areas that are under- or over-exposed.

As part of this research, an experimental HDR camera system was taken to film a launch in July 2013. As the dynamic range in a rocket launch is a lot brighter than 14 stops, two Canon 5D Mark IIIs, chosen for the availability of the Magic Lantern firmware, were mounted "bottom-to-bottom" using a special mount, as seen in Figure 6.1a. As can be seen, rubber foam was placed between the lenses to minimise vibrations. An additional challenge was that, for safety reasons, the capture team was 3 miles from the launch site. A 400mm lens together with a $2\times$ focal length extender was used for each camera. Figure 6.1b shows the capture in progress, while Figure 6.1c shows a tone mapped frame of the launch. The Canon 5D Mark IIIs, each using (different) alternative exposures, gave a combined dynamic range of approximately 18 stops. This was unfortunately not enough to fully capture all the detail in the flume of the rocket as well as the surrounding scene.

(a) Camera system used (b) Rocket launch (c) Frame (tone mapped for printing) from the launch

Figure 6.1 – Images from the case study.

6.1.1.2 Second generation HDR video capture system

The experience with the rocket launch clearly showed (a) the need to be able to capture as wide a dynamic range as possible, to ensure all the detail in the scene was subsequently available and (b) the limitations of a two-camera solution. In particular, the two Canon 5D Mark III system, was cumbersome, easily moved out of alignment, and introduced parallax effects.

A second-generation commodity system was thus constructed using a small form factor SDI camera, the Flare 2KSDI, shown in Figure 6.2. The chief advantage of this camera over the Canon system was that the capture parameters could be programmed directly over a serial interface, avoiding the dependency on specialised firmware and allowing the capture software to set the exposure controls itself. This removed the need for user intervention and gave a "hands-off" HDR camera solution. In addition, the 10-bit SDI interface available from the camera delivered more dynamic range to the merging algorithm, providing only a three-exposure merge. The single HDR-enabled Flare system was measured as being capable of capturing 18.2 stops by the Kennedy Space Center.

6.1.2 Real-time HDR video compression

In order to be able to compress and stream HDR video using commodity hardware a traditional HDR video encoding method was adapted and accelerated to give the performance required. The goHDR compression method [96] is well suited for acceleration as it exploits existing high performance video encoders. The goHDR method, being a two-stream method, splits the HDR video frame into 2×8-bit frames which can then each be fed into optimised CPU

Figure 6.2 – The second-generation HDR system, implemented here with an FPGA to enable remote deployment.

based video encoders such as x264 [114] or even take advantage of embedded hardware video encoders present on many modern GPUs and in the SoCs (system of chips) present in high-end mobile phones. Another advantage of using legacy video codecs is the ability to move the compressed HDR video data through existing broadcast pipelines as well as via robust streaming solutions such as HLS (HTTP Live Streaming) and RTSP (Real Time Streaming Protocol). This means that the HDR data can be streamed over the internet using existing systems in order to be displayed on any device, including mobiles.

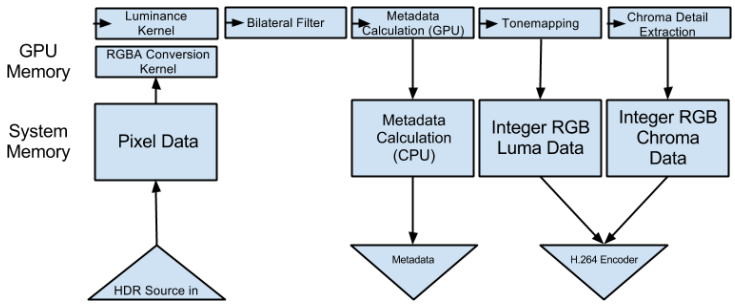


Figure 6.3 – goHDR-classic HDR video compression method system architecture.

Figure 6.3 shows the block diagram for the system architecture used for goHDR-classic. The method was originally designed for CPU computation and therefore a number of operations needed to be adapted for use in a parallel, GPU based implementation, in order for real-time compression and streaming to be achieved. As Figure 6.3 shows, most of the method proved suitable to pixel-independent processing using GPU kernels. As Figure 6.4 shows, the bulk of the time spent on the GPU was taken up by the bilateral filter, which makes multiple image

memory reads, and so the bulk of the optimisation effort was spent on the bilateral and averaging stages of execution.

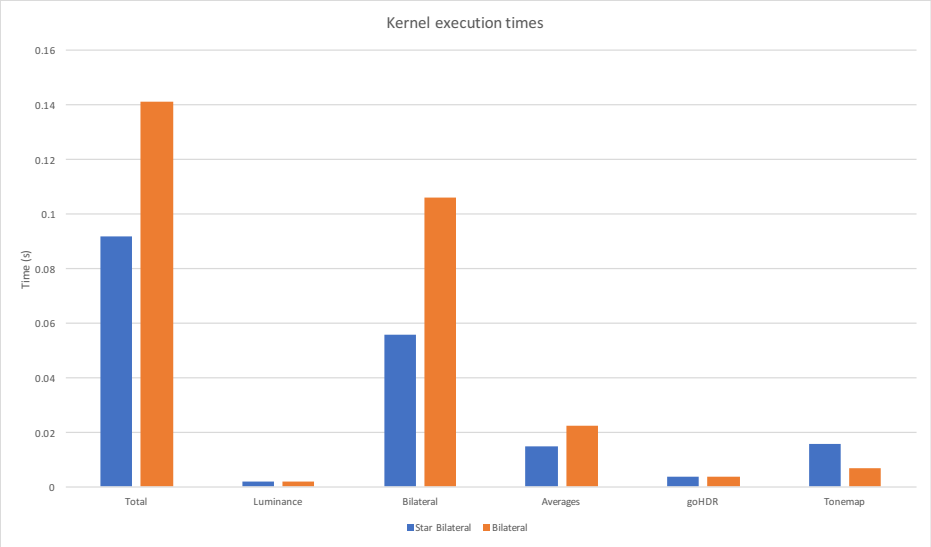


Figure 6.4 – Preliminary average kernel timings over 150 runs of the different stages of processing in the goHDR method, highlighting the major contribution of the bilateral stage.

6.1.2.1 Pixel data intake

Pixel data is taken from the input source and transmitted directly to the GPU as quickly as possible, making use of the OpenCL 'pinned memory' technique for direct transfer from system memory to GPU memory.

6.1.2.2 RGBA conversion kernel

The HDR RGB 32-bit floating point data was expanded to RGBA data for improved performance via a novel kernel which packs three channel image data into a four channel buffer, then on a parallelisable basis expands it to the native four channel format required by OpenCL. While RGB data can be converted at speeds up to real-time on the CPU, this method provided a significant reduction in overall system latency.

6.1.2.3 Luminance kernel

The luminance was computed via per-pixel convolution with the Rec. 601 luminance primaries as follows:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{6.1}$$

6.1.2.4 Bilateral filter

Subsequently an optimised 'star' bilateral filter was applied, as show in Figure 6.6. This star bilateral filter, which operates in a series of decomposed loops, is an optimised combination of the separable bilateral filter and the box bilateral filter [35, 115], as shown adapted for the GPU in Figure 6.5. It is designed to scale linearly with the size of the sample window instead of quadratically, as shown in Figure 6.7. The slight loss in precision, < 0.5 PSNR in compressed footage was considered acceptable for the $\approx$ 82% average gain in performance (see Table 6.3) which enables pipeline to achieve real-time performance. A variant of the bilateral filter was tested which utilised OpenCL local memory, but proved inferior to the spatial caching performed by the OpenCL image types.

6.1.2.5 Metadata generation

The filtered luminance is used to compute a number of statistics from the frame to aid compression. These include the minimum, maximum, average and harmonic mean values. In order to efficiently calculate single values for each frame a mip-mapping, pyramid-like scheme was used as shown in Figure 6.7c. Such a scheme removes the need for a costly copy of the image from GPU memory back to system memory and also avoids the use of GPU atomic variables which are a significant performance bottleneck. The code was originally tested with the final iterations, where the image being downsampled would be small, being transferred to the CPU. This proved less efficient than remaining on the GPU and only reading out the final result, as shown in Figure 6.3

```
__kernel void bilateral(read_only image2d_t input, write_only image2d_t
    output, float sigmaS, float sigmaR, int halfWindow) {
  const sampler_t sampler = CLK_FILTER_NEAREST
  | CLK_NORMALIZED_COORDS_FALSE
  | CLK_ADDRESS_CLAMP_TO_EDGE;
  const int2 size = get_image_dim(input); // Image size
  int2 q, p = (int2)(get_global_id(0), get_global_id(1));
  if (p.x <= size.x && p.y <= size.y) {
    const float ip = read_imagef(input, sampler, p).x;
    float sum = 0.0f, norm = 0.0f;
    const int2 start = (int2)(p.x - halfWindow, p.y - halfWindow), end =
        (int2)(p.x + halfWindow, p.y + halfWindow);
    for (q.y = start.y; q.y <= end.y; ++q.y) {
      for (q.x = start.x; q.x <= end.x; ++q.x) {
        if (q.x >= 0 && q.y >= 0 && q.x <= size.x && q.y <= size.y) {
          float iq = read_imagef(input, sampler, q).x;
          float2 space = (float2)(p.x - q.x, p.y - q.y);
          float gs = exp(-sigmaS * (space.x*space.x + space.y*space.y));
          float range = ip - iq;
          float gr = exp(-sigmaR * (range*range));
          float factor = gs * gr;
          sum += factor * iq;
          norm += factor;
        }
      }
    }
    const float out = sum / norm;
    write_imagef(output, p, (float4)(out, out, out, out));
  }
}
```

Figure 6.5 – A standard bilateral filter, implemented in OpenCL.

```
__kernel void star_bilateral(read_only image2d_t input, write_only image2d_t
    output, const float sigmaS, const float sigmaR, const int halfWindow) {
  const sampler_t sampler = CLK_FILTER_NEAREST | CLK_NORMALIZED_COORDS_FALSE
  | CLK_ADDRESS_CLAMP_TO_EDGE; // An unnormalised OpenCL image sampler.
  const int2 size = get_image_dim(input); // Image size
  int2 q, p = (int2)(get_global_id(0), get_global_id(1));
  if (p.x <= size.x && p.y <= size.y) {
    float ip = read_imagef(input, sampler, p).x;
    float sum = 0.0f, norm = 0.0f;
    sum += ip;
    norm += 1.0f;
    q = p;
    for (q.x = p.x - halfWindow; q.x <= p.x + halfWindow; ++q.x) {
      if (q.x >= 0 && q.x <= size.x && q.x != p.x) {
        float iq = read_imagef(input, sampler, q).x;
        float2 space = (float2)(p.x - q.x, p.y - q.y);
        float gs = exp(-sigmaS * (space.x*space.x + space.y*space.y));
        float range = ip - iq;
        float gr = exp(-sigmaR * (range*range));
        float factor = gs * gr;
        sum += factor * iq;
        norm += factor;
      }
    }
```

Figure 6.6 – The star bilateral convolution kernel. sigmaS and sigmaR are sigma variables following the bilateral filter as in Tomasi and Banduchi [35]. halfWindow is a half window size.

```
    q = p;
    for (q.y = p.y - halfWindow; q.y <= p.y + halfWindow; ++q.y) {
      if (q.y >= 0 && q.y <= size.y && q.y != p.y) {
        float iq = read_imagef(input, sampler, q).x;
        float2 space = (float2)(p.x - q.x, p.y - q.y);
        float gs = exp(-sigmaS * (space.x*space.x + space.y*space.y));
        float range = ip - iq;
        float gr = exp(-sigmaR * (range*range));
        float factor = gs * gr;
        sum += factor * iq;
        norm += factor;
      }
    }
    for (q.x = p.x - halfWindow, q.y = p.y - halfWindow; q.x <= p.x +
        halfWindow; ++q.x, ++q.y) {
      if (q.x >= 0 && q.y >= 0 && q.x <= size.x && q.y <= size.y && q.x !=
          p.x) {
        float iq = read_imagef(input, sampler, q).x;
        float2 space = (float2)(p.x - q.x, p.y - q.y);
        float gs = exp(-sigmaS * (space.x*space.x + space.y*space.y));
        float range = ip - iq;
        float gr = exp(-sigmaR * (range*range));
        float factor = gs * gr;
        sum += factor * iq;
        norm += factor;
      }
    }
```

Figure 6.6 – The star bilateral convolution kernel (cont).

```
    for (q.x = p.x + halfWindow, q.y = p.y - halfWindow; q.x >= p.x -
        halfWindow; --q.x, ++q.y) {
      if (q.x >= 0 && q.y >= 0 && q.x <= size.x && q.y <= size.y && q.x !=
          p.x) {
        float iq = read_imagef(input, sampler, q).x;
        float2 space = (float2)(p.x - q.x, p.y - q.y);
        float gs = exp(-sigmaS * (space.x*space.x + space.y*space.y));
        float range = ip - iq;
        float gr = exp(-sigmaR * (range*range));
        float factor = gs * gr;
        sum += factor * iq;
        norm += factor;
      }
    }
    float out = sum / norm;
    write_imagef(output, p, (float4)(out, out, out, out));
  }
}
```

Figure 6.6 – The star bilateral convolution kernel (cont).

6.1.2.6 Sigmoid tonemapper

The filtered luminance is then tone mapped using an invertible tone mapper [76] adapted for storage, rather than for a particular display characteristic, in order to store the full range of luminance in the frame.

6.1.2.7 Chroma encoding

The chrominance and residual luminance samples are then passed into a second 8-bit integer image [78].

6.1.2.8 Passing data from GPU to a codec

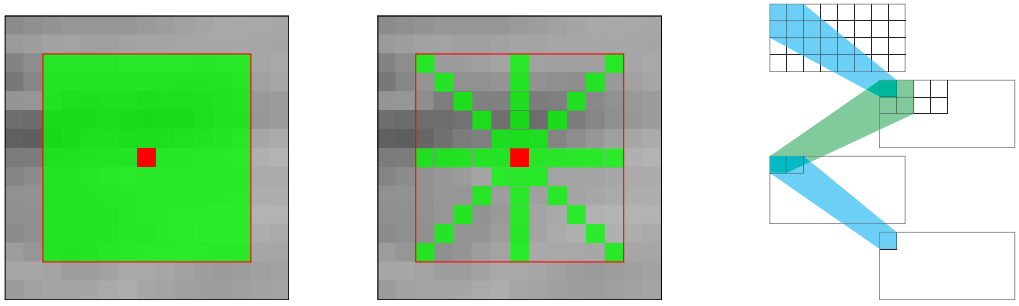
The two frames are then withdrawn from the CPU and converted from 32-bit floating point to 8-bit integers with a clamp, then are passed to two individual video encoders, in this case H.264, for compression. They can then be transmitted either as two discrete video streams in a single container, or else as a double-sized frame.

6.1.2.9 Implementation

The GPU implementation was written in OpenCL as the implementation is cross-platform. This has the key advantage that the method can be ported to other platforms, including FPGAs and mobile devices in a straightforward manner. Each stage in Figure 6.3 was implemented as a single OpenCL kernel. The blocks which remain in system memory are data transfers back down to system memory, for transmission or storage.

6.1.3 HDR video on commercially available displays

As discussed earlier, although commercial HDR displays such as the SIM2 are available [74], for the foreseeable future the majority of HDR video content will be viewed on LDR displays, including mobile devices and emerging 10-bit LDR displays capable of up to 1,000 cd/m^2 peak luminance. A dedicated HDR Video Player was thus developed to provide an enhanced viewing experience of HDR content even on such LDR displays. Capable of running on a wide range of platforms, including Android and iOS mobile devices, this player (see Figure 6.8) has the following functionality:



(a) Box bilateral filter with half window size of 5.

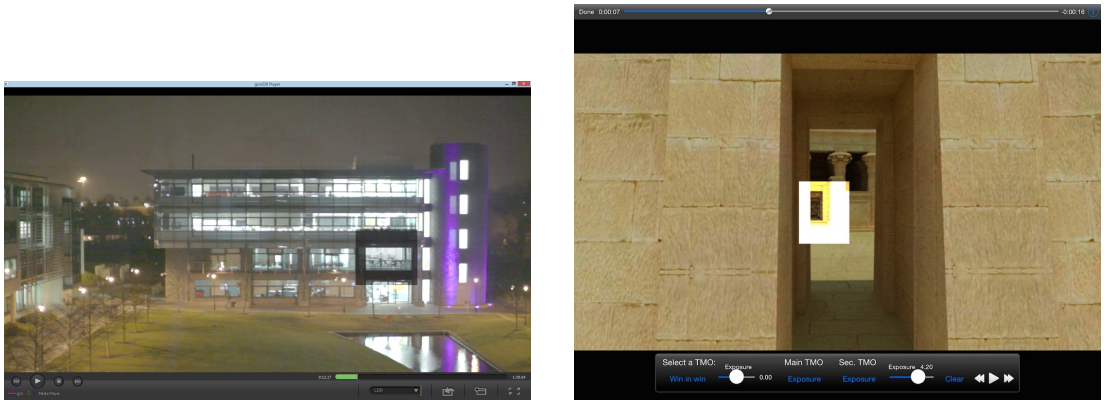
(b) Star bilateral filter with half window size of 5.

(c) Optimised method of image statistic calculation based on mip-mapping.

Figure 6.7 – (a) & (b) The box and star methods of bilateral filtering showing sampled pixels in green surrounding the filter target in red. (c) Using this method multiple statistics can be generated for the entire frame in a parallel fashion.

- Support for a wide range of LDR and HDR video formats and image sequences.
- The ability to select the display device as LDR, SIM2 or newer 10-bit displays with an HDR transfer function.
- A choice of tone mapping operators to suit display and content, including a logarithmic operator, the filmic TMO [116], and the Drago TMO [117]. Melo et al. proposed a methodology for establishing the ideal TMO for a given viewing scenario [?].
- The ability to select and display an exposure of the HDR scene.
- Windows-in-windows can be included and examined at different exposure values as shown in Figure 6.8a and Figure 6.8b.

The player thus provides a viewer the opportunity for an interactive personalised experience. If the dynamic range of the delivered footage is larger than that of the display a dynamic choice of tone mappers can be selected, or the viewer can explore scene detail in the different exposures, including setting an appropriate exposure for an object or part of the scene to ensure that the associated detail can be clearly seen. This will allow, for example, the viewer to see their favourite player, no matter what the lighting conditions of the actual scene are, or as Figure 6.8a shows,



(a) HDR Video Player for Windows.

(b) HDR Video Player for iOS.

Figure 6.8 – HDR Video Player. A separate window is created to enable detail to be seen simultaneously in both the interior and exterior of the building.

the ability to simultaneously watch a person in a lit room of a building at night as well as one on the outside.

6.2 Methodology

6.2.1 Scenes used

Six HDR video sequences of 150 frames in the uncompressed PFM format were chosen as the test footage for this experiement. The sequences, shown as thumbnails in Table 6.2, were selected to represent a range of HDR video test cases:

- **CGRoom**: Fully CG footage of a barrel falling off a shelf in a dark room with a single swinging bulb.
- **Welding**: Indoor footage shot using the Spheron HDRcam, a prototype HDR camera utilising beam splitters. The footage depicts arc welding in a high-contrast environment.
- **Tears of Steel** [118]: A mix of live-action and CG footage made available by the Blender Foundation.
- **Seine** [119]: Professional-quality footage provided by Technicolor for HDR evaluations. The footage depicts a riverboat hung with bulbs moving down the river Seine.

- **Jaguar**: Slow footage, shot using a dolly, showing the interior and exterior of a classic car.
- **Mercedes**: An outdoor pan past a car in direct sunlight.

Except where cited, the footage was produced internally.

6.2.2 Test hardware

The results were calculated using an Intel i5-3570K CPU and a Geforce GTX 750 Ti graphics card, to demonstrate the performance of mid-range commodity computing hardware.

6.3 Results

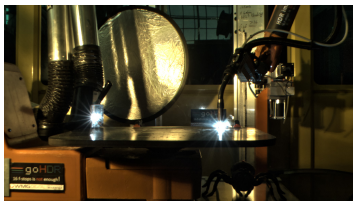
This section presents the performance results obtained with our commodity end-to-end HDR video pipeline.

For real-time performance in a pipeline the requirement is for each stage of processing to introduce as little latency as possible. At the very least, the latency must equate to a processing frame-rate equal or greater than the target rate of the video (typically between 24 and 30 frames per second). Any improvement above that figure will reduce the overall latency of the system but is not essential.

In Table 6.3a the latency incurred by the proposed HDR video pipeline described above is measured, both running on a CPU and subsequently a GPU. Furthermore, the method is evaluated separately for both the box bilateral and star bilateral filters described above, as well as a control evaluation in which no filtering is performed. From these results it is clear that the bilateral filtering represents the bulk of the processing time in the method, and that the use of the star bilateral filter allows real-time 30fps performance to be achieved when a GPU is used. These results are also presented in terms of overall frame-rate in Figure 6.9. This graph shows that, as expected, performance measured between sequences is roughly equivalent for sequences of the same resolution. The Tears of Steel sequence, which has a smaller resolution, has faster results.

Figure 6.10 shows an average calculation of the PSNR metric under the three filter options in the previous graph, from the results in Table 6.3a. The quality of the image output from the HDR encoding section of the system is shown to be minimally affected by the choice of filter.

Table 6.2 – HDR video sequences used for evaluation. Each sequence consisted of 150 frames.

	Name	Image	Resolution	Dynamic Range (Stops)
(a)	CGRoom		1920 × 1080	27.81
(b)	Welding		1920 × 1080	25.4
(c)	Tears of Steel [118]		1920 × 800	24.9
(d)	Seine [119]		1920 × 1080	25.2
(e)	Jaguar		1920 × 1080	30.6
(f)	Mercedes		1920 × 1080	30.6

As no subsequent video encoding occurs in this test system, the benefit of the bilateral filter to the process overall is not demonstrated.

Figure 6.11 demonstrates that the schema detailed in Figure 6.7c for using the GPU to calculate image metadata (average, maximum, minimum, harmonic mean) produces a performance improvement, although even with the CPU schema (for averaging, with the bilateral filtering still performed on the GPU) the system still provides performance of 25fps.

Table 6.3 – Per-frame latency incurred in HDR encoding with box bilateral versus star bilateral implemented on GPU (a) and CPU (b), and the penalty incurred to quality (c).

(a) CPU Latency (Seconds)

	CGRoom	Welding	Tears of Steel	Seine	Jaguar	Mercedes
Box Bilateral	0.624	0.647	0.482	0.650	0.735	0.735
Star Bilateral	0.252	0.250	0.210	0.277	0.284	0.277
No Filter	0.072	0.073	0.079	0.101	0.099	0.097

(b) GPU Latency (Seconds)

	CGRoom	Welding	Tears of Steel	Seine	Jaguar	Mercedes
Box Bilateral	0.045	0.045	0.036	0.049	0.049	0.048
Star Bilateral	0.026	0.025	0.021	0.025	0.026	0.025
No Filter	0.013	0.013	0.010	0.013	0.014	0.013

(c) PSNR (dB)

	CGRoom	Welding	Tears of Steel	Seine	Jaguar	Mercedes
Box Bilateral	60.75	53.25	40.35	61.54	51.66	53.70
Star Bilateral	60.73	52.99	40.30	61.17	51.24	53.42
No Filter	60.92	54.93	39.81	60.47	50.32	51.98

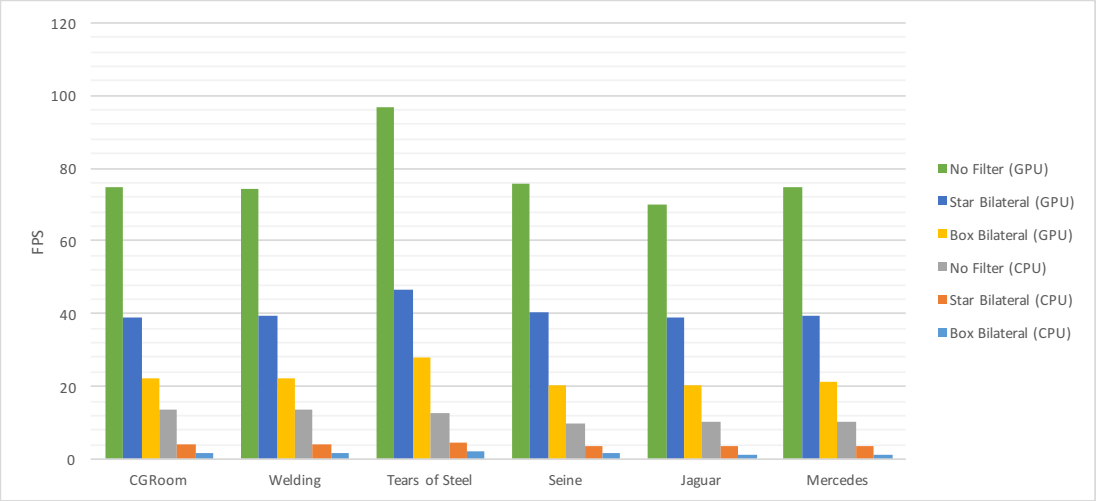


Figure 6.9 – Performance comparison of Star and Box Bilateral filters on CPU and GPU.

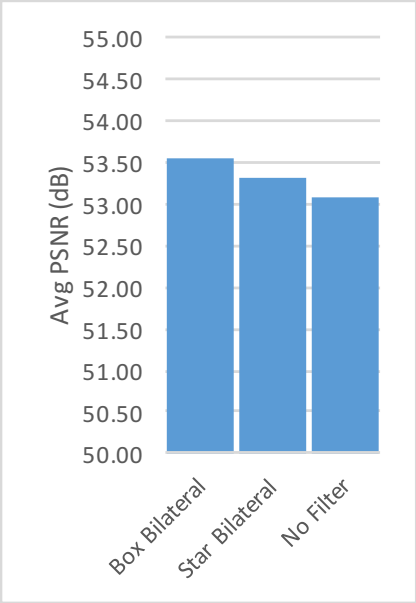


Figure 6.10 – Average PSNR achieved using the Box and optimised Star Bilateral filters.

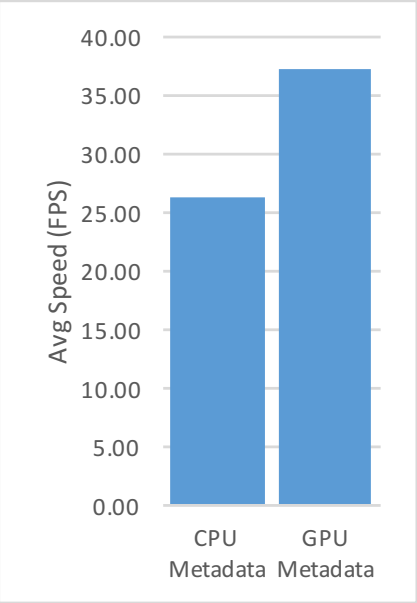


Figure 6.11 – Performance increase gained from moving frame metadata calculation from CPU to GPU.

6.4 Summary

To achieve the full potential of the step change in image quality that HDR video provides, it must be possible for a wide range of the population to be able to capture and display their own HDR video content. State-of-the-art cameras, such as the ARRI Alexa and the Red Epic, which can directly capture a wide dynamic range, and HDR displays, such as the SIM2 are out of the price range of most consumers. This chapter has presented a complete end-to-end HDR pipeline using off-the-shelf hardware. This integrated pipeline can include everything from capture to post-production, archival and storage, compression, transmission, and display in a straightforward manner. As the results show, it is possible to achieve real-time performance along this entire pipeline and deliver the full range of HDR data to any display.

Chapter 7

Streaming High Dynamic Range Graphics

> *In simple optics, the ray directed from the radiating point to the illuminated point arrives by the shortest direct path in the same medium.*
>
> — Gottfried Leibniz, *Unicum Opticae, Catoptricae et Dioptricae Principium*

Video streaming for HDR is naturally of interest within the context of remote rendering. The rationale for this is that rendered graphics, especially high fidelity rendered graphics, are typically generated in HDR, even if they are subsequently tone-mapped either for display or for transmission. The nature of calculating the transportation of light within a scene is such that accurate lighting information even beyond that which is presentable on the user's display is needed in order to compute subtle lighting effects such as reflections and refractions.

Remote rendering allows resources, such as supercomputers, computing clusters or cloud-based computing providers, to be utilised from anywhere where there is a suitable internet connection and a baseline capable client.

A key advantage of HDR CG content, compared to HDR content captured by an imaging device (as discussed in Chapter 6), is that additional information regarding the generated content is available at the rendering engine. This chapter investigates the exploitation of this additional data, including knowledge of depth, geometry and motion to further increase the performance of a state-of-the-art HDR video compression algorithm in the streaming of re-

mote rendered graphics. Results produced show the significant benefits that such an approach can achieve for remote rendering compared to compressing the generated HDR video stream without these enhancements.

7.1 Rendering metadata

A rendering engine used to produce an image via ray tracing or some other high-fidelity graphics production method necessarily has access to useful per-pixel metadata about the scene which the image was produced from.

7.1.1 Depth map

Specifically, information about the objects in the scene can be accessed by producing a depth map, in which each pixel value is the length of the ray cast to the primary object intersected.

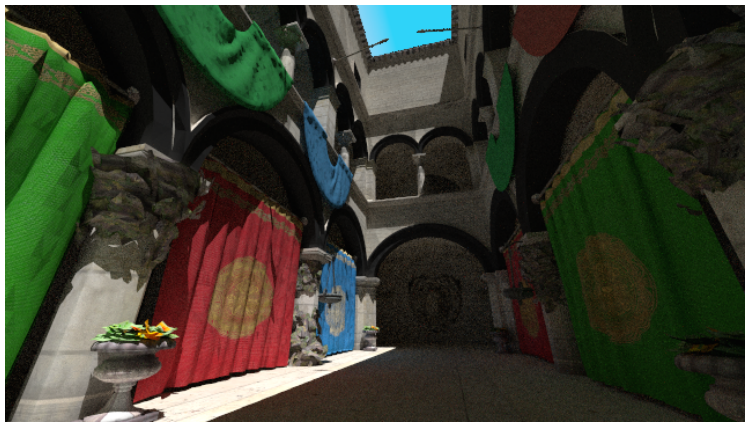
7.1.2 Geometry buffer

Also easily available is normal map, which encodes information about the geometry of the scene into a per-pixel format by taking the normal of the object at the point of intersection.

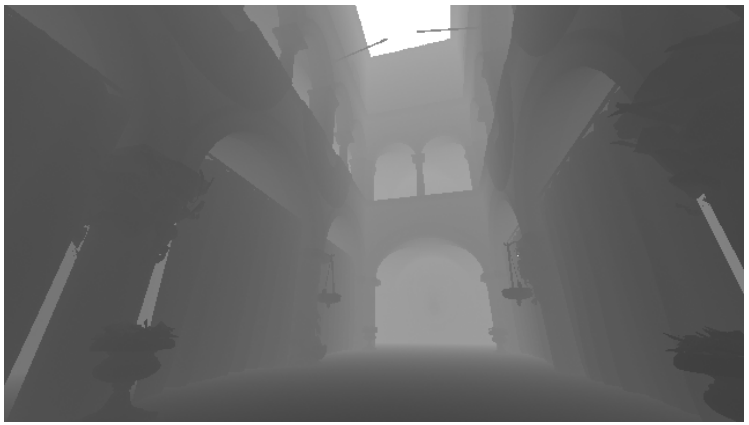
7.1.3 Motion flow

Slightly more complex, a mapping of scene motion can be produced by taking each of the points of intersection and casting a ray back from the object not to the current camera, but to the position of the previous camera. This gives coordinates, in screen-space, of the currently displayed location in the previous rendered image. This can then be encoded as a difference, producing an image of motion vectors: a warping from the previous image into the current one.

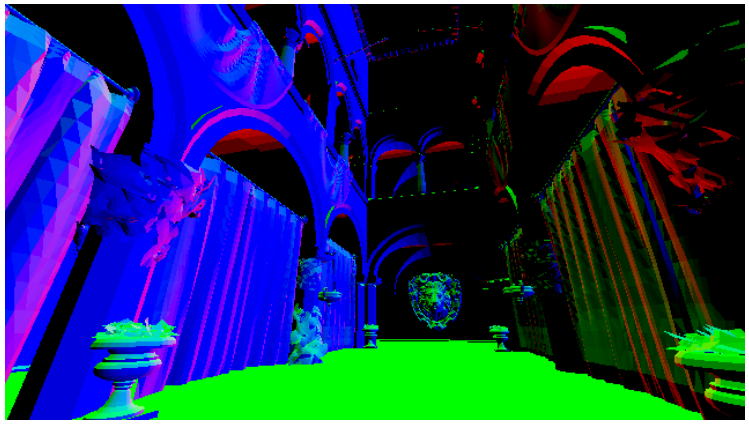
An issue occurs where objects are occluded or disoccluded within the previous or current image. In this case, the warping will no longer be accurate to the objects currently in view. To counteract this, it is necessary to encode with the motion vectors the length of the ray cast back to the previous camera position; this information can be compared with a depth map of the currently viewed scene, and where the depths are not within a certain tolerance (for this chapter, the tolerance is < 0.1) the warped information can be discarded.



(a) Rendered image.



(b) Depth map.



(c) Geometry buffer.

(d) Motion flow.

Figure 7.1 – The types of metadata considered.

7.2 Streaming metadata

Following Pajak et al. [39], in this chapter a method of remote augmented streaming developed for rasterised graphics is adapted to high-fidelity path tracing rendering of HDR content.

The graphics renderer is adapted to produce, alongside a full-resolution graphical output, several metadata buffers which are easily acquired from its regular function: depth, surface normal, and motion flow. The renderer used is a path tracer and thus depth and surface normals are produced as part of the ray tracing process.

Motion flow was generated by re-projecting the object coordinates backwards to the previous camera position, adding one more ray per sample.

These metadata buffers are then each taken through a compression schema. In this, edge samples are taken using a Laplacian kernel and the results are thresholded to a particular level of detail. These sparse samples are then separated into a binary map and a gapless block of data values. The binary map is split into four interleaved images.

Each is subsequently encoded as a difference from the previous interpolated map using the JBIG2 binary map encoder. The data samples are encoded, first with a generic predictor and then with entropy coding.

Additionally, a regular set of samples is taken from the original buffer and this low resolution image is encoded with the Paeth predictor [63] and entropy coding.

These extra regular samples aid in preserving gentle gradients within the metadata buffers. These three frames, thus packaged, constitute the compressed metadata buffers.

7.2.1 Downsampled rendered image

At the same time, the full-resolution graphical output from the renderer is downsampled to a quarter of its original size and encoded in a lossy fashion by a standard video encoder, for example H.264. This video frame is packed alongside the compressed metadata buffers in a standard container and streamed using standard video streaming methods.

At the client end, the bundle of information is unpacked into the video frame and the metadata. The video frame is decoded by a standard hardware or software video decoder, while the metadata is unpacked and the lossless aspects of the compression (binary encoding, pre-

diction) are reversed. Left with a sparse set of samples, a push-pull upsampler [64] is used to reconstitute these samples into a full image.

7.2.2 Push/pull expansion of edge images

The metadata images thus arrive at the client in the form of a compressed image composed of regular samples, which is depended on for representing gradients and other gentle changes in value, a blob of packed data representing the edge samples, and a binary map in JBIG2 format showing their location in the image. To transform this information back into a full frame image, a push-pull method is used. With the image of regular samples decoded and the edge sample map brought out of JBIG2 and assigned its values from the packed data, this information is placed into a full sized image at its corresponding pixel locations, with the edge samples (which are more reliable) prioritised over the regular samples.

The image is then pushed downwards repeated, with the data following a fixed schema for prioritising detail as described in Chapter 3. In addition, the Laplacian sign data, which was encoded into the packed edge samples, is used to pick preference for contiguous edge blocks. The pushes are limited to five steps, at which point - by virtue of the regular samples taken every five pixels, at this point the downsized image cannot have any further gaps. At this point, the image is 'pulled' back up, with gaps in the larger images being filled with interpolated data from the smaller images, with respect to the authoritative edge samples, until a full metadata image is produced.

7.2.3 Adaptation to Path Tracing

The method as described in Pajak et al. [39] is prescribed for use with raster video engines, where performance can be directly linked to shader passes over pixels and both depth and normal buffers can be cheaply calculated. It is posited that the method could be adapted for path tracing systems but the details of such an implementation are not provided.

7.2.4 Metadata-based upsampling

The rendered image frame, downsized for transmission, is now intelligently upscaled using a fast multi-dimensional filter to scale the image [40]. In this method a multi-dimensional bilat-

eral filter is used to perform an upscale which prioritises depth discontinuities and the edges of objects in motion, via the depth metadata image and the motion metadata image. These discontinuities are the most salient parts of each frame and, by keeping their edges at a high quality, the quality of the final image is perceptually improved. Naïve upsampling techniques have to compensate for their inability to know where definite edges by detecting edge algorithmically, as with the simple bilateral filter. As the edges are known to the renderer, this can be avoided.

7.2.5 Reprojection

As our images are generated from a renderer rather than a camera, we have access to accurate information about the image which is unavailable in other circumstances. We can quickly and cheaply generate metadata about the scene from its geometry, such as depth, and from depth, screen-space motion vectors. For the goal of generating stereo imagery, the left eye position can be projected to the right eye position. The process of reprojection can be seen in Figure 7.2, which shows an input image, two depth maps and a reprojected output. As well, the final image in Figure 7.2 shows a regular samples of points in the original image with a line drawn to their relocation in the second image, with a split between the two images in the background.

7.3 Adapted real-time response via spatial-temporal interpolation

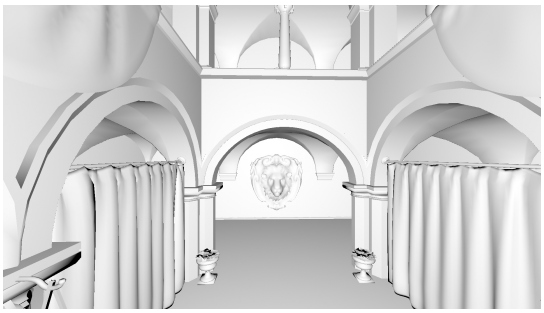
A further development of this method utilises the temporal 'history buffer' as described by Herzog et al. [38], but instead of taking advantage of this by distributing rendering spatially over time, it is re-purposed to provide continuity to a real-time path tracer.

A temporal component is thus introduced, which uses the motion flow to reproject the previously received rendered frame into the current camera position.

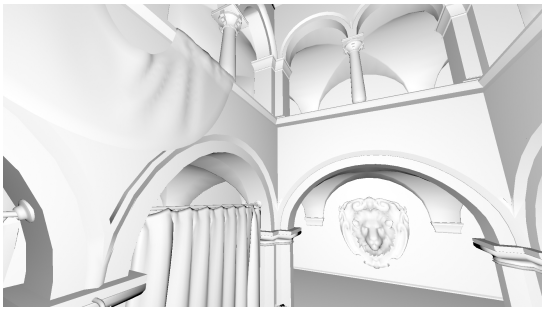
This 'predicted' frame then receives a weighted blending with the incoming output from the ray tracer. The motion flow produced at the server end is provided with a sample of the reprojected camera depth, to allow areas of the previous image being obscured by motion in the current image to be disregarded in the reprojection.

This sequential blending of previous and current frames allows continuity from frame-to-frame in a rendering system which otherwise starts each frame afresh. Because of the weighting, the output frame will (absent motion) converge to the correct output quickly over time (in our

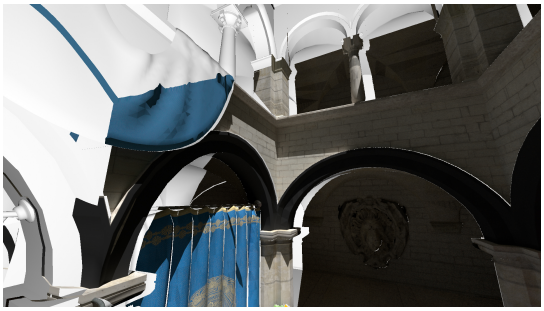
(a) Initial image



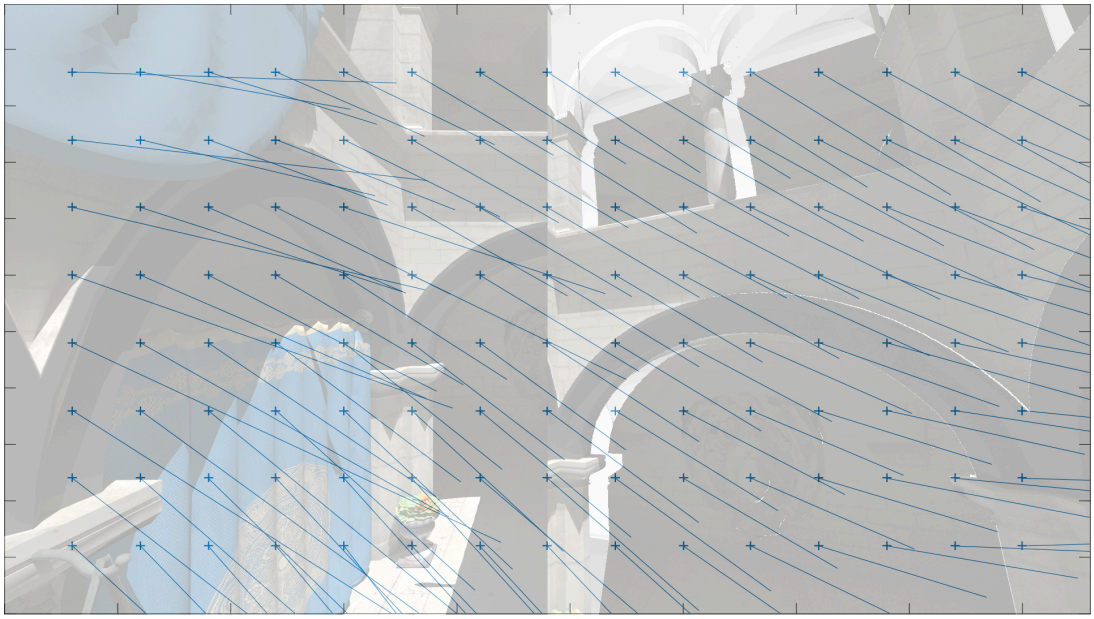
(b) Unshaded raster representation.



(c) Unshaded raster second perspective.



(d) Pixels from (a) reprojected.



(e) Vector diagram showing projection. The left side of the image is overlaid with the source, the right side the destination.

Figure 7.2 – The reprojection process.

system, 10 frames).

This method of blending a reprojected frame in to the received data has a necessary cost in terms of objective metrics because the image it produces, while presenting a more continuous output to the user, is not strictly related to the frames being output by the renderer for comparison.

7.3.1 Adaptation for HDR

High dynamic range imagery is inherent to the process of path tracing. As such, this makes it an ideal fit for emerging one-stream standards of streaming HDR video, such as the SMPTE-2084 standard. HDR information, delivered to the client, can afford client-side benefits such as examining the details in both bright and dark regions of the rendered image without any further server-side processing, or automatically adapting the displayed brightness of the image to ambient lighting conditions. In addition, streaming this remotely rendered HDR CG content to the client enables it to be displayed directly on an HDR display, such as a SIM2 HDR47 [74], or tone mapped for showing on a Standard Dynamic Range (SDR) display.

To be adapted for streaming HDR content, the method described in this chapter was implemented in the GPGPU-based image processing environment described in Chapter 6. This permitted the easy replacement by configuration of the H.264-based video streaming component, which used the Rec709 gamma curve, with a solution for 10-bit H.265 streaming. This stream was encoded using the SMPTE-2084 PQ-curve based transfer function and decoded accordingly for display.

7.4 Methodology

One of the goals of this work was to assess the benefits of metadata-augmented streaming techniques for delivering real-time path tracing from a remote renderer to a client. As access to such a real-time path tracing rendering engine was not readily available, the real-time aspect was simulated by pre-generating a set of input and output frames.

As well, to support the HDR aspect of the experiment, the results were saved in the near-lossless high dynamic range image format OpenEXR [121]. For each of 150 frames of a pre-computed motion path through the scene, a high-resolution rendered graphic was obtained

(over 1,000 samples per pixel), as well as three sets of high-resolution metadata.

This information could then be played back in real time directly into the GPU-based implementation of the method, as well as the regular H.264 and H.265 encoders. The results were subsequently decoded within the same system, and metrics obtained at that point.

To assess the efficiency and any quality implications of our approach, we obtained results for latency encountered in the method, network bandwidth used, a rough indication of hardware load on server and client, and quality as measured with a number of objective metrics.

For SDR video imagery, the most commonly used quality metric used is Peak Signal-to-Noise Ratio (PSNR) [122]. In addition, there is SSIM [123], the structural similarity index. For HDR video imagery there is also the Visual Difference Predictor, a metric for estimating subjective perception of image quality [124] and its High Dynamic Range extension HDR-VDP [125], as well as the HDR Video Quality Metric, HDR-VQM [126].

For the purposes of this experiment, PSNR and SSIM will be compared, as well as select HDR-VDP results to verify the preservation of HDR information in the transmitted data.

These metrics will be used to compare the quality of a high-resolution locally rendered image on the server (in ideal conditions), with that of received and decoded images presented to the user at the client end.

7.4.1 Scenes used

In this experiment, results are presented for the Sponza [127], Sibenik [128] and Conference Room [129] scenes.

- **Sponza** [127]: The atrium of the Sponza palace in Dubrovnik. A standard test scene, with a moderate triangle count and objects situated in the near-field to demonstrate occlusion.
- **Sibenik** [128]: Another classic test sequence, the model of the Sibenik cathedral in Croatia was originally created by Marko Dabrovic. It is the simplest model tested.
- **Conference room** [129]: This model of a small conference room originally created by Anat Grynberg and Greg Ward has chairs and a table, representing a potential use for remote rendering, virtual teleconferencing.

  A sample image from each scene is shown in Table 7.1.

Table 7.1 – The scenes used in the HDR reprojection experiment.

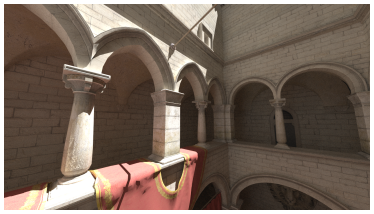
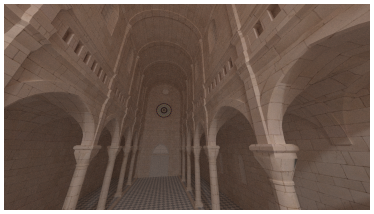
	Name	Image	Triangles	Dynamic Range (Stops)
(a)	Sponza		262267	24.9
(b)	Sibenik		75284	30.3
(c)	Conference Room		331179	22.3

Table 7.2 – PSNR results for each sequence.

	PSNR (dB)			
Scene	Full H.264	Low-res H.264	Spatial	Spatial-temporal
Sponza	34.0226	32.7399	32.5447	31.6858
Sibenik	43.2588	41.7473	41.771	41.3099
Conference Room	28.7481	28.7481	28.7482	28.7482

7.4.2 Test hardware

The encoding methods were run on commodity hardware, a Geforce 750ti and an Intel 3.5GHz i5 processor.

7.5 Results

Results are presented here in Table 7.2 and Table 7.3 for two variations on the current method, one which incorporates a temporal element (2) and one which does not (1), compared against a baseline low-resolution H.264 stream encoded at CRF 34 and a gold-standard high-resolution stream encoded at CRF 14. Presently the results demonstrate a small advantage for the method in SSIM and a small negative for PSNR. This is potentially due to the smoothing effect of the method on noisy path-traced renderings adversely affecting the PSNR result.

Also presented in Table 7.4 and Table 7.5 are results comparing the method adapted for HDR H.265 10-bit encoding with SMPTE-2084 [98], in PSNR and HDR-VDP. These are compared similarly with equivalent gold-standard and baseline video streams, and the metrics are taken against a full dynamic range original floating-point image. Using the HDR coding, while the spatial method holds its own, the temporal-temporal method suffers worse than the default. This is likely due to artefacts in very bright regions being more noticeable than in standard dynamic range.

Table 7.3 – SSIM results for each sequence.

	SSIM Index			
Scene	Full H.264	Low-res H.264	Spatial	Spatial-temporal
Sponza	0.901165	0.891977	0.894993	0.881935
Sibenik	0.915061	0.914842	0.917378	0.907700
Conference Room	0.886375	0.883953	0.886070	0.873631

Table 7.4 – PSNR results for each sequence under the HDR encoder.

	PSNR (dB)			
Scene	Full H.265	Low-res H.265	Spatial	Spatial-temporal
Sponza	33.7903	32.1379	32.0791	31.3727
Sibenik	42.8083	41.2067	41.2639	40.8814
Conference Room	31.7182	38.2634	38.1678	37.6882

Table 7.5 – HDR-VDP quality results for each sequence (0-100) the HDR encoder.

	VDP			
Scene	Full H.265	Low-res H.265	Spatial	Spatial-temporal
Sponza	58.2735	56.6332	56.2676	52.4972
Sibenik	73.3377	63.9327	64.4665	61.5108
Conference Room	46.5797	46.1361	45.2338	43.1079

7.6 Summary

This chapter presented an efficient method for remote rendering of high-fidelity, HDR graphics, based on the augmentation of a low-resolution video stream with metadata buffers. This solution was shown to be usefully effective for both standard video encoding and incoming high dynamic range video encoding methods. In the following chapter, the concepts presented here, most notably reprojection, will be taken advantage of to provide a low-latency stereo effect for virtual reality headsets.

Chapter 8

Remote rendering for virtual reality

> *It is inevitable that [the spectacle] should elevate the human sense of sight to the special place once occupied by touch; the most abstract of the senses, and the most easily deceived...*
>
> — Guy Debord, *The Society of the Spectacle*

This chapter explores methods of efficiently streaming graphics rendered in high quality so as to supply an HMD or any other virtual reality display with minimal latency. Specifically, this chapter presents novel results related to the use of equirectangular projections for virtual, rendered experiences, based on reprojection from calculated motion flow.

This is accomplished, in part, by dividing the work between client and server with the bulk of the rendering processed on the high performance resource and the client resources being used to provide an immediate response to user input. Depth data computed locally is used to permit the client to respond to user head movement without dependence on the server.

These techniques for presenting additional views depend on a single full-quality transmitted frame and varying quantities of additional data. As the client has both a high-quality 360° image to pan and tilt within, as well as access to depth information with which to reproject that image to simulate spatial motion, applications can be kept responsive over increased server latency.

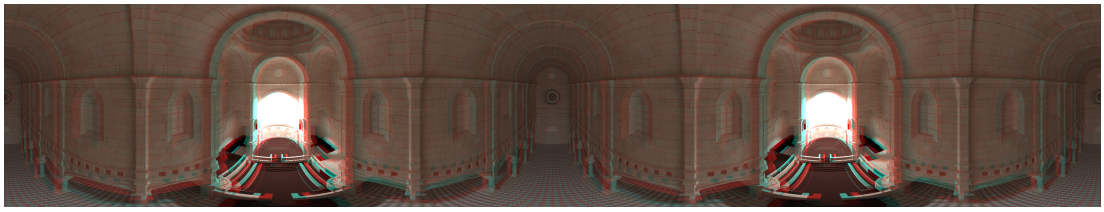


Figure 8.1 – An anaglyph stereo representation of an equirectangular projection of the Sibenik scene. Viewed with anaglyph glasses, the stereo effect can be observed varying over the flow of the image, as the viewer's eyeline relates to the stereo positioning of the 360° cameras.

8.1 Overview

This chapter compares an existing stereo remote rendering system using reprojection to generate a stereo pair, with an innovative system which streams 360° projections to allow free changes of viewpoint within a single frame. In this way the difficultly of tracking the head movement of a user in a HMD under bandwidth and rendering time restrictions is reduced, lowering the overall latency of the system.

8.1.1 Equirectangular projection

The process proposed uses the equirectangular projection for all the 360° maps within this chapter. This decision was made due to the ease of integration of this mapping into existing video streaming pipelines and the ease with which it can be integrated into high-fidelity renderers. A further consideration was the adoption of the equirectangular projection as a de facto standard for the streaming of 360° video by online video services such as YouTube [131] and Facebook [132].

The task of adapting a ray tracer to generating an equirectangular projection is fairly straightforward: Rather than the flat, screen-shaped camera plane used in traditional ray tracing, instead a set of rays are generated based on spherical coordinates to map out a full sphere. These rays are then evaluated in the typical fashion and the resulting colour values (and any relevant metadata) can be mapped back into a regular image. The x axis in the image becomes panning rotation around the sphere's y axis and the y axis in the image serves as vertical tilt, applied to the camera direction C_{dir}. In Equation 8.1, i and j are normalised image coordinates and R_x

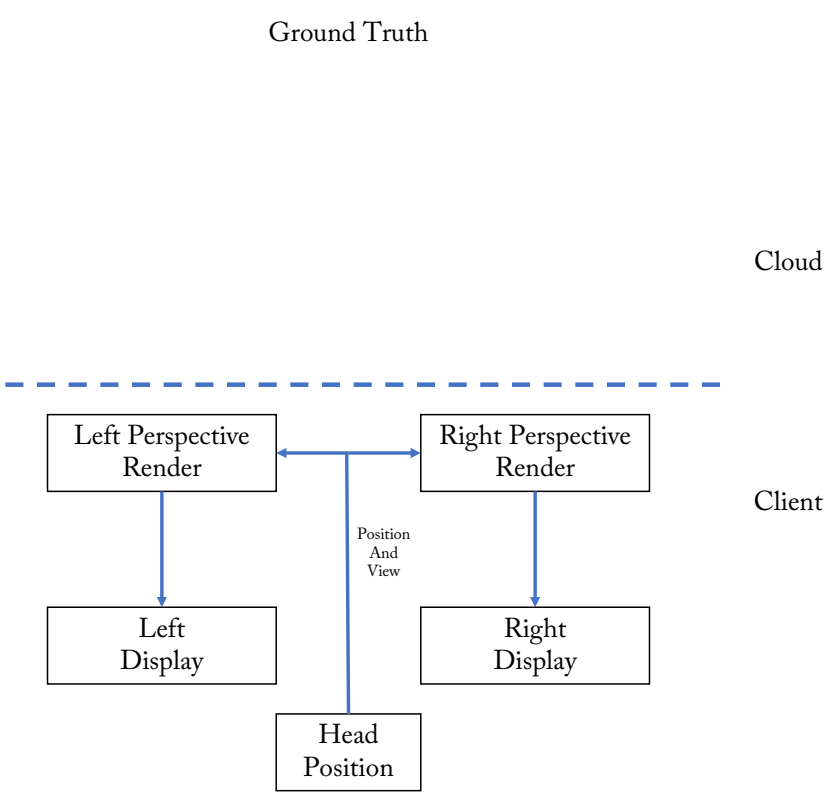


Figure 8.2 – The ground truth process for rendering stereo: two perspective images, both generated locally.

and R_y are 3D rotation matrices for in x and y, defined as a function on radians. The ray is thus composed of the camera position, and a camera direction vector transformed by a composed rotation matrix based on the normalised coordinates.

$$Ray(i,j) = C_{pos} + R_x(-\frac{\pi}{4} + \frac{\pi}{2} \cdot j) \cdot R_y(\pi \cdot i) \cdot C_{dir} \tag{8.1}$$

In order to display a regular perspective image to the user, the client must be capable of mapping the equirectangular projection to a sphere within a 3D environment. With this performed, any pure rotation of the HMD can be simulated as rotation of the view of the sphere. In addition, a small amount of realism can be gained by having displacement represented as movement within the sphere – e.g. having leaning forward bringing the user's perspective closer to the surface of the sphere. This is a very limited effect however, as the inability of the projection to reveal occluded locations will quickly become apparent.

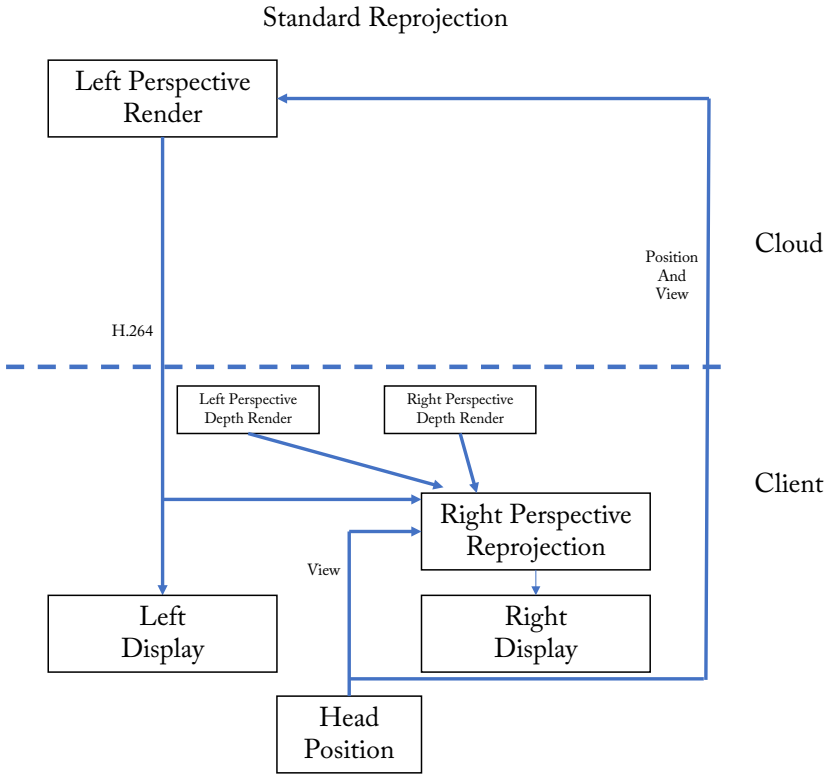


Figure 8.3 – An existing reprojection-based remote rendering stereo solution. a single image is rendered remotely, encoded with H.264, and transmitted to the client where it is reprojected to provide a stereo perspective.

8.1.2 Reprojection

As considered in the spatial-temporal method in Chapter 7, the acquisition of more than one view from a single frame is reprojection, where the image values of a previous image temporally or spatially are projected back to a new camera position. This provides a screen-space transformation from one image to the other, with only disoccluded regions, i.e. gaps, remaining unaccounted for.

8.1.3 Reprojecting equirectangular images

Reprojection using equirectangular images works similarly to reprojecting a perspective image, in that the projection of the camera is inverted to produce a line back to the initial camera position. An inherent advantage with a spherical camera is that as a full 360° sweep is included

in each image, the problem specific to a perspective rendering, that the spatial distance between the two eyes causes necessary gaps at the edge of the image does not occur.

8.1.4 Generating depth images

Where traditional methods of reprojection have relied on using a depth channel sent alongside a rendered video, this chapter evaluates an innovative method of generating depth maps in real time using a raster engine and a copy of the local geometry data. In this way, the client can respond to increases in latency from the server by continuing to form reprojections of the current and previous images received, to allow seamless continuity of view movement for a user until the server recovers. These depth maps are used for both the standard perspective method as well as a raster equirectangular projection for the equirectangular method.

In scenarios where the geometry of the scene is too large or too resource-intensive for the client to responsively process, a H.264 or H.265 stream of the depth buffer can be transmitted from the server alongside the processed images, at a reduced quality [44].

8.1.4.1 Client reprojection using a raster engine

To achieve minimal latency, it is essential that for any given image, the client is capable of reprojecting it to suit changes in orientation and position immediately. The round trip time to the remote renderer to obtain the depth maps may prove unsuitable.

The raster engine needs access to the same model which the remote renderer uses, but does not need to perform resource-intensive tasks such as loading textures or generating lighting models. To achieve this effect, certain features of the raster engine must be exploited: a high precision Z-buffer needs to be used, and the model must be of sufficient detail either by pre-processing or via GPU tessellation such that the interpolation which is characteristic of raster engines is not so inaccurate that the motion vectors generated are unsuitable.

Once an accurate depth buffer has been obtained for the new camera position, the motion flow generation can occur in the same manner as in the remote renderer, by ray projection back to the previous camera position. For this to work, the client must maintain a record of camera position by tracking the changes it reports back to the remote renderer. This process is illustrated in Figure 7.2.

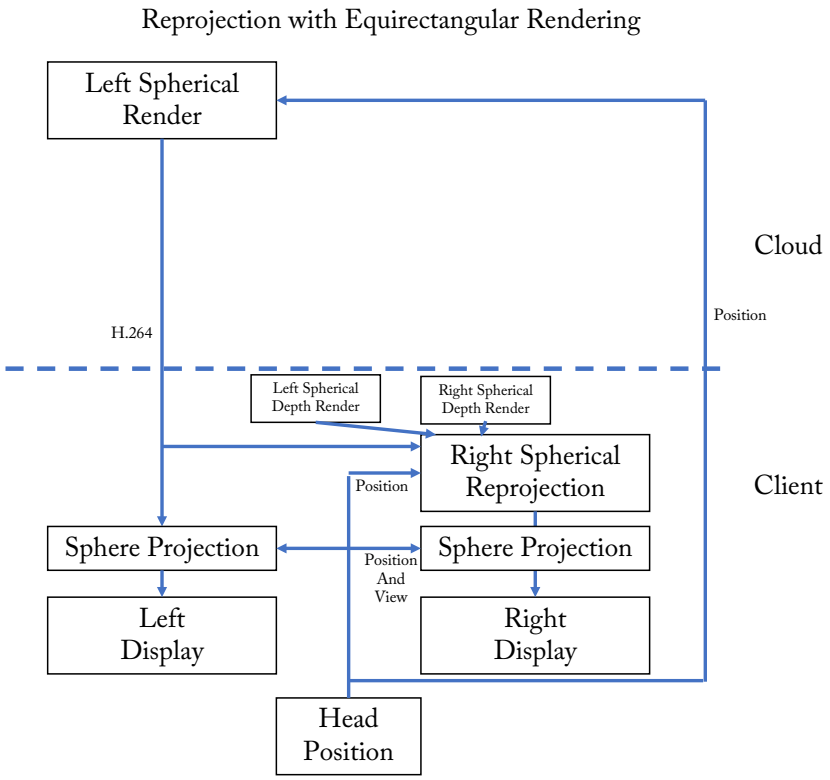


Figure 8.4 – The proposed system of remote rendering, in which a single equirectangular projection is rendered remotely, encoded with H.264, then transmitted to the client, where it is transformed to provide two separate perspective views.

This method could also be adapted to generating stereo pairs for HMDs and 3D monitors. It should be noted however that much consumer hardware for 3D display depends on making its own adjustments to the raster engine rendering pathway, and does not permit the use of individual generated stereo image pairs. HMDs are a happy exception to this rule.

8.1.5 Head mounted displays

A major challenge in the display of virtual reality via high quality graphics methods such as path tracing is that, even more than with traditional human-computer interfaces, the HMD must be constantly updating to respond to unconscious or conscious head movements by the user. This tracking information is typically provided from HMD sensors in the form of a displacement vector and a matrix transformation to represent 3D head rotation [133], and as an individual

is very rarely absolutely still the values are changing continuously. Furthermore, HMDs also require an image for both eyes, and as well as the two images needing to be appropriately displaced to generate the stereo effect, the images must also update at the same time to avoid causing nausea [9].

Even the most violent directional changes can be compensated for interactively if a full 360° field of view has been rendered. So long as the head remains in one location, even as the gaze moves, no further action is required of the rendering engine. It is the job of the client to take slices of the received image map and present them as perspective images to the user.

A downside to this approach is that compared to transmitting a full perspective image, for a single frame the quality of the image will be worse. This is because necessarily many of the pixels generated for a 360° mapping will be located behind the camera.

As well, generation of a stereo pair becomes a problem because while a single sphere map can remain static within the sphere for any gaze position, to achieve the stereo effect a sphere for the second eye must maintain the appropriate relative position to the first sphere, moving in an orbit around it. This greatly limits the potential to reduce rendering caused by head movements as the map for the second eye is not helped. This can be observed in Figure 8.1.

A general issue with 360° projections is that, due to the difficultly of accurately mapping the surface of a sphere to a rectangular projection, the level of detail (LoD) across the image is mixed, with an overallocation of pixels to the regions at the poles of the sphere compared to the equator. In addition, artefacts are possible at the boundaries of the texture where the pixels are no longer contiguous.

8.1.6 Reprojection at client

To achieve reprojection at the client, this extra information must also be available at the client. Motion flow information can be transmitted as a pair of single-channel depth images, one for the image to be projected to (which permits the reprojection to be calculated) and one for the original frame (so that by difference, disocclusions can be registered where the calculated depth does not match the depth of the reprojection. As well, motion flow can be transmitted pre-calculated, as a two-dimensional screen mapping and a binary map of disocclusions.

To achieve even greater efficiency, if the client is capable of efficiently generating the motion

flow itself via a raster engine and a local copy of the scene vertex data, then the reprojection can be achieved without having to transmit depth or motion information at all. This requires the client to have knowledge of the camera position and scene geometry.

Disocclusions, in the case where they are not too prominent, can be handled by rendering a low-quality image to be sampled from just in the case of disocclusions. This image can be generated either remotely and streamed, or via the graphics capabilities of the client.

Where reprojection becomes difficult is in the case where either disocclusions are too prominent to be replaced satisfactorily with a low quality alternative. The subtle head movements tracked by an HMD suggest reprojection to be not just used for generating a stereo pair for the second eye, but also to compensate for head position. Rapid head movements however, such as a look over the shoulder, can have very little in common with previous frames and thus very little to reproject.

8.1.7 Reprojection and 360° mapping

This chapter proposes ameliorating the downsides of both of these methods of efficiently streaming to a stereo HMD by combining them. Reprojection presents a fast path for taking a static sphere map and quickly reprojecting it to different second eye positions, and both the lack of movement in the sphere map and the nature of a 360° field of view, changing little from position to position, make reprojection an attractive prospect for rendering to a HMD.

8.2 Methodology

For the results presented in this chapter, a set of image sequences were rendered in high fidelity, then passed through three methods for remote rendering in both LDR and HDR, and the quality of the resulting images was measured against the bandwidth required to transmit them. Bandwidth-per-pixel is directly related to latency, in that a real-time system which requires low transmission of data overall can be assumed to be low-latency.

The three methods tested were a ground truth transmission of both frames encoded, a reprojection system where a left eye perspective image was reprojected to provide a right eye, and an equirectangular projection system where a left equirectangular projection is reprojected to provide a right equirectangular projection, which are both then remapped to the view of the

user.

8.2.1 Image set generation

Five sequences of frames were rendered using an unbiased path tracer at 1,000 samples per pixel for 150 frames, at 1080p resolution. The same resolution was used for both perspective renderings and equirectangular renderings, in order to place the same demand on the server for both image sets. The sequences followed a predefined camera path which was also available to the client to simulate client-side movement of a user. Each sequence was rendered using a path tracing renderer at both stereo viewpoints as well as rendering dual 360° equirectangular projections using a spherical camera, one at each stereo eye position.

8.2.2 Metrics

To compare the image quality of the outputs, two standard image metrics, PSNR and SSIM, were used [122, 123]. Both metrics were used to compare outputted images to the 'ground truth' provided by the original renders for the left and right stereo perspectives. PSNR measures the loss of fine detail in the images, while SSIM measures the structural similarity. PSNR is expressed in dB, with higher values representing greater fidelity to the original image, while SSIM is expressed as an average index over the image, with values closer to 1 representing high fidelity.

For the HDR outputs, the metrics used are puPSNR (perceptually uniform PSNR) and puSSIM (perceptually uniform SSIM), adaptations of PSNR and SSIM to HDR images [134].

8.2.3 Filling gaps

Gaps introduced in reprojection can be handled with a number of techniques, including interpolation or filling with a simple raster representation of the scene calculated on the client. Reprojection methods typically struggle with reflection and refraction due to the complicated way reflections and refractions move along with camera motion. Regions where this causes significant disparity between ground truth and the reprojections can be treated as gaps.

For this experiment, where the reprojection technique introduced gaps into the image these were filled using a 16x smaller render of the relevant stereo image encoded in a second video

stream encoded alongside the relevant left eye image, at the same CRF.

8.2.4 Scenes used

Five scenes were tested, Sponza [127], San Miguel [135], Rungholt [135], Kitchen and Sibenik [135]. The sequences are shown as thumbnails in Table 8.1. The models were chosen to represent a range of environments likely to be encountered in virtual reality scenarios, including indoor and outdoor scenes as well as more and less visually complex scenes. The San Miguel and Rungholt models have a high triangle count to test performance on larger models.

- **Sponza** [127]: The atrium of the Sponza palace in Dubrovnik. A standard test scene, with a moderate triangle count and objects situated in the near-field to demonstrate occlusion.
- **San Miguel** [135]: Originally modeled by Guillermo M. Leal Llaguno, this scene is based on a hacienda in San Miguel de Allende, Mexico. It has the highest polygon count of the models used, much of it concentrated in thick foliage.
- **Rungholt** [135]: A converted Minecraft model, the Rungholt city was chosen as it is a large, open model with plenty of sky and direct lighting.
- **Kitchen**: A model of a simple kitchen living space to demonstrate domestic VR.
- **Sibenik** [128]: Another classic test sequence, the model of the Sibenik cathedral in Croatia was originally created by Marko Dabrovic. It is the simplest model tested.

8.2.5 Encoder

For the LDR results, the rendered frame sequences were encoded using x254, an H.264 encoder. The videos were encoded at seven quality levels, CRFs 4, 8, 12, 16, 20, 24 and 28, to characterise the interaction of the method with the encoder over a wide range of typical average bitrates.

For the HDR results, the rendered frame sequences were encoded in 10-bit using x265, an H.265 encoder. The videos were encoded at four quality levels, CRFs 8, 16, 24 and 28.

8.2.6 Test hardware

The server-side path tracing was preprocessed using a render farm and the ground truth input to the methods was provided using full-range OpenEXR files. The client side reprojections were

performed using a Macbook Pro with a 2.6ghz Intel Core i7 and a Geforce 750M GPU. The code was implemented for the GPU using a combination of OpenGL for the client-side depth renderings and OpenCL for the HDR processing, using the framework described in Chapter 6.

8.2.7 Generating depth images

In this experiment, using an OpenGL raster graphics engine, a high precision Z-buffer was achieved using glClipControl (a core feature in OpenGL 4.5 [136]) to enable a 0 to 1 range floating point depth buffer, as opposed to the OpenGL standard -1 to 1 range. Combined with the reverse Z-buffer method [137], in which the near plane of the camera is mapped to 1 and the far plane (in this case infinity) is mapped to 0, this produces results which suffer little to no loss of accuracy [138].

The equirectangular projection depth images were generated by doing six perspective renders onto the sides of an OpenGL cube map and performing a custom shader which transformed the cube map into the correct projection.

8.2.8 Pipeline

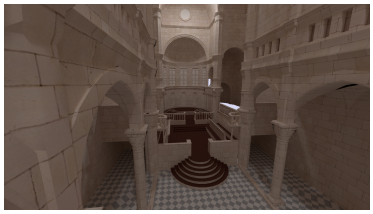
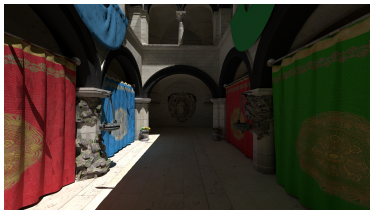
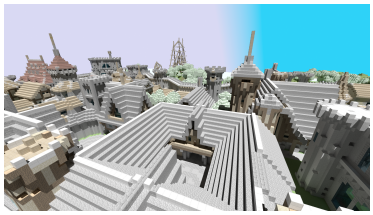
To obtain the results, the rendered frame sequences were encoded with the relevant video encoder and curve, then decoded into EXR files, then reprojected and/or projected onto a sphere, using the original camera path to maintain direction of view. In this way, pixel-accurate representations of the same view were obtained for both the reprojections and the equirectangular projections, which could be compared using quality metrics to the ground truth.

8.3 Results

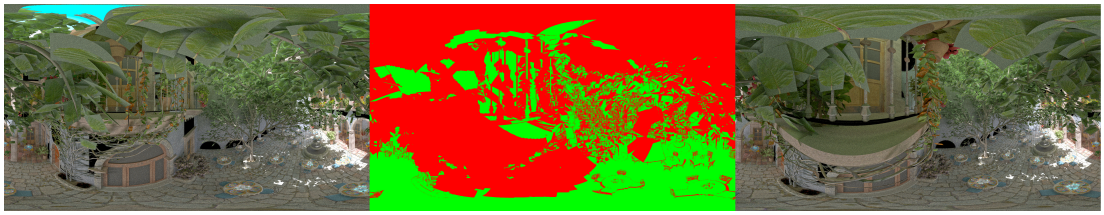
The objective metric results for LDR are presented as rate-distortion graphs for PSNR in Figure 8.6, Figure 8.7 and Figure 8.8. Performance is shown first for the left eye image, then the right eye image, then as an average of the two to give an overall performance metric. Rate-distortion graphs are also presented for SSIM in Figure 8.9, Figure 8.10 and Figure 8.11, with the same layout.

Similarly, the objective metrics for HDR are presented as rate-distortion graphs for puPSNR

Table 8.1 – The rendered sequences used for the 360° reprojection experiment.

	Name	Image	Triangles	Dynamic Range (Stops)
(a)	Sibenik		75284	30.3
(b)	Kitchen		134130	25.4
(c)	Sponza		262267	24.9
(d)	Rungholt		6704264	25.2
(e)	San Miguel		7852950	30.6

(a) Perspective



(b) Spherical

Figure 8.5 – A single frame from the San Miguel sequence, in both perspective and spherical renderings. In the centre image, pixels from the right image visible in the left are shaded green. Pixels occluded are shaded red. In both perspective and spherical views, objects close to the camera disrupt the potential for reprojection.

in Figure 8.13, Figure 8.14 and Figure 8.15, and puSSIM in Figure 8.16, Figure 8.17 and Figure 8.18.

Across all the graphs, it can be seen that the measured quality of the images tends to drop as more transformations are made from the original rendered output to the final presented image, from regular encode to reprojection to equirectangular reprojection. For scenes featuring simpler texturing, such as the Kitchen scene, the effect is less pronounced. The drop is also less pronounced in the higher CRF images, indicating that the loss of quality is to do with high frequency information being lost by reprojection, which as it is derived from resampling rather than the original image source, has no relation to the noise level in the original image.

The generation of the image sets at the same resolution necessarily puts the equirectangular images at a quality disadvantage, as a high percentage of pixels rendered aren't displayed in any one projection onto a sphere. The rate-distortion graphs in LDR and HDR for the equirectangular projection show that a quality ceiling is quickly reached for the equirectangular projections, where the addition of more bits per pixel does not provide a corresponding increase in quality. This restriction is assumed to be a limit of the image resolution.

In both the LDR and HDR result sets, for the Rungholt, Kitchen and Sibenik scenes the standard reprojection methods outperform the ground truth at low bits per pixel. These demonstrate the benefits of a locally-generated depth map-based reprojection approach.

In Figure 8.12 and Figure 8.19 it can be seen that the processing times for both methods are similar for three of the five scenes, but in the two high-polygon outdoor scenes (San Miguel and Rungholt) the required client-side processing is greater. This is due to the increased utilisation of the raster engine providing depth images when calculating the equirectangular depth maps, which are formed of six individual cube face renders, requiring six passes of raster rendering. The two models affected are the high-triangle scenes, San Miguel and Rungholt. It is possible that through more aggressive model culling these figures could be brought back to the level of the rest of the scenes. There is no significant difference between the processing time in LDR versus HDR for the client.

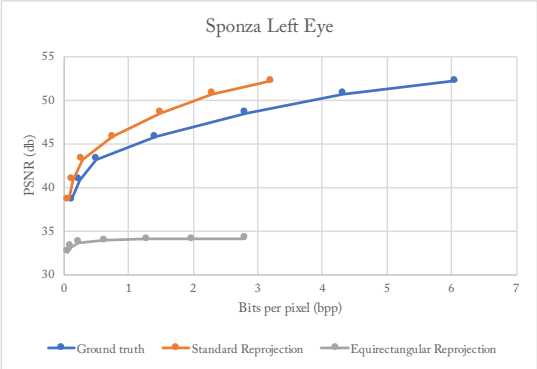
8.3.1 San Miguel

The San Miguel sequence suffered in particular for all reprojections as can be seen in Figure 8.10, as in the data set the left eye is briefly obstructed several times by leaves in the scenes. This effect can be seen in Figure 8.5 for both perspective and spherical images. As the leaves close to the camera lens varying obscure different areas of the background almost at random, there is providing little to no valid data with which to reproject the left eye into the right eye position. In such a circumstance, the system falls back to an upscaling of the low-resolution stream from the server.

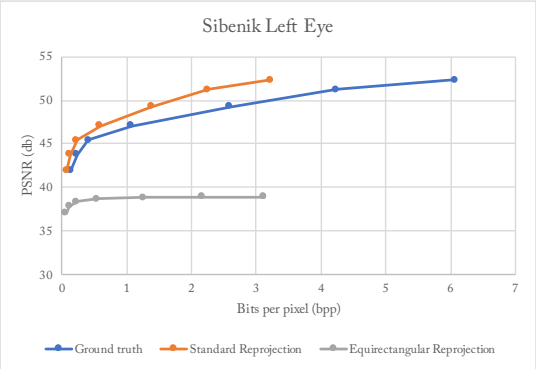
This situation could be detected through summing the number of depth test failures as in the centre column of Figure 8.5. A future system might, in this circumstance, rank the most useful candidate frame from both temporal and spatial reprojections by comparing depth values. The temporal reprojection from the same is more likely to be useful in this scenario, where the leaf is approached up until it obscures the viewport.

8.4 Summary

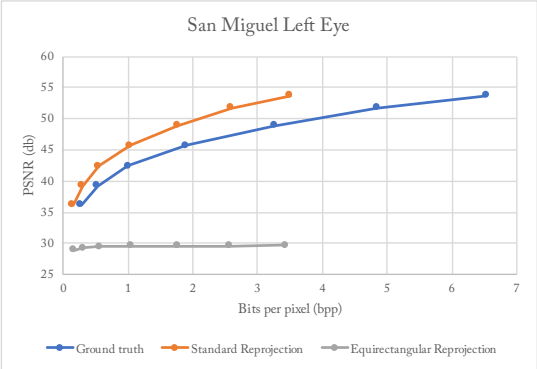
This chapter presented a method for streaming high-fidelity rendered graphics for virtual reality with minimal latency. The results, related to the use of equirectangular projections for path



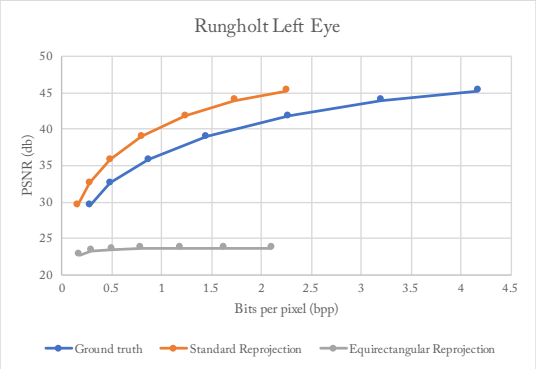
(a) Sponza



(b) Sibenik



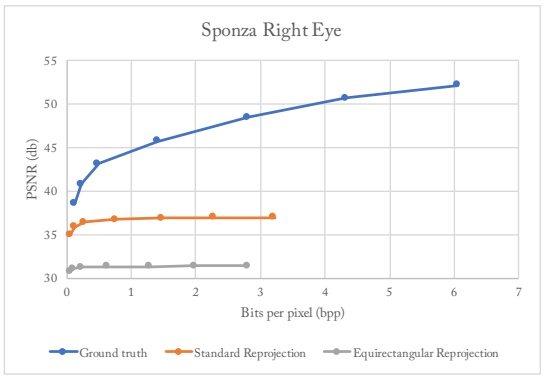
(c) San Miguel



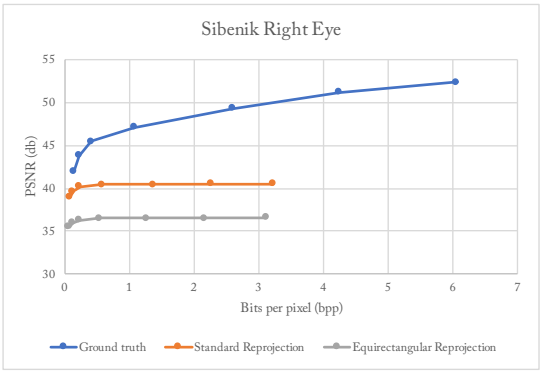
(d) Rungholt

(e) Kitchen

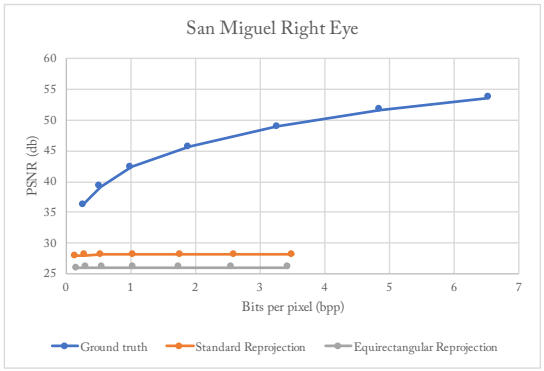
Figure 8.6 – Rate-distortion graphs for LDR streaming of the left eye over three methods (PSNR).



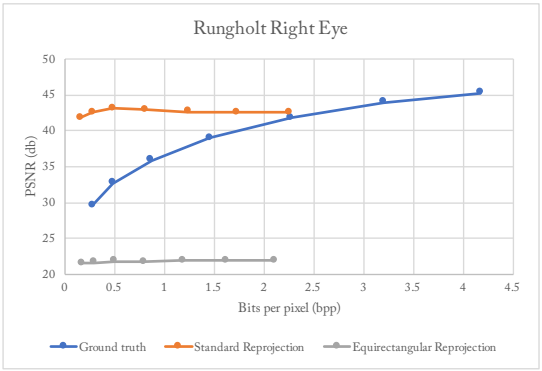
(a) Sponza



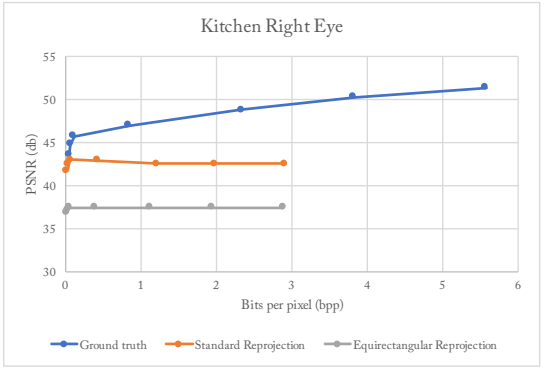
(b) Sibenik



(c) San Miguel

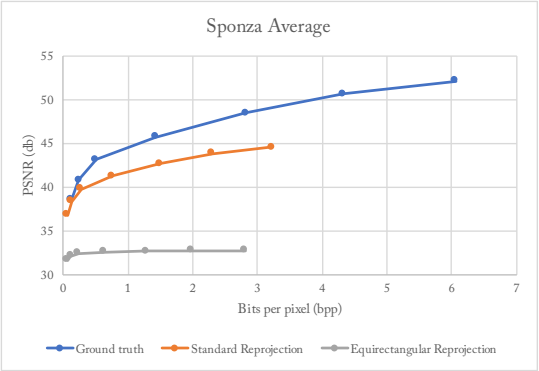


(d) Rungholt

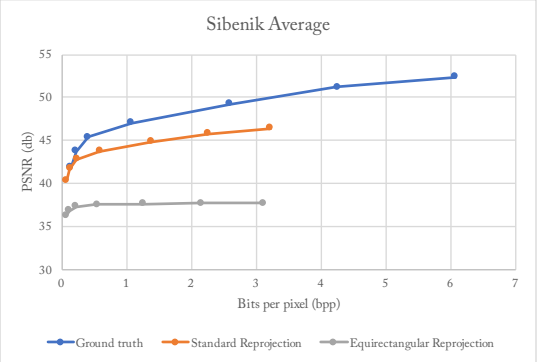


(e) Kitchen

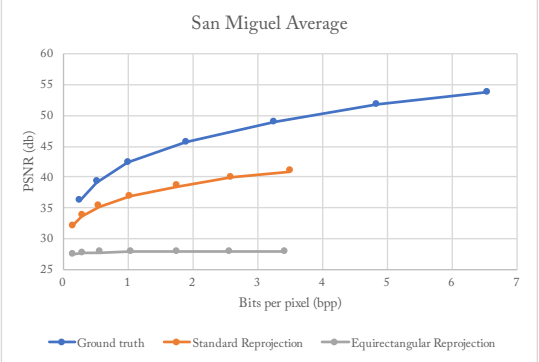
Figure 8.7 – Rate-distortion graphs for LDR streaming of the right eye over three methods (PSNR).



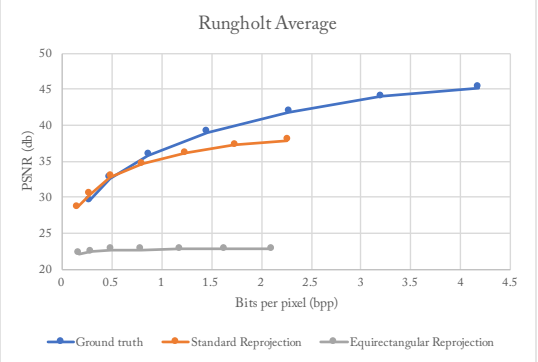
(a) Sponza



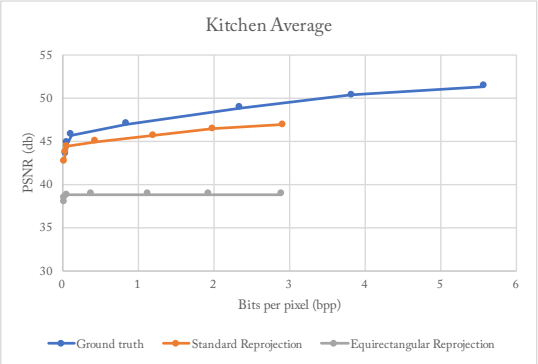
(b) Sibenik



(c) San Miguel



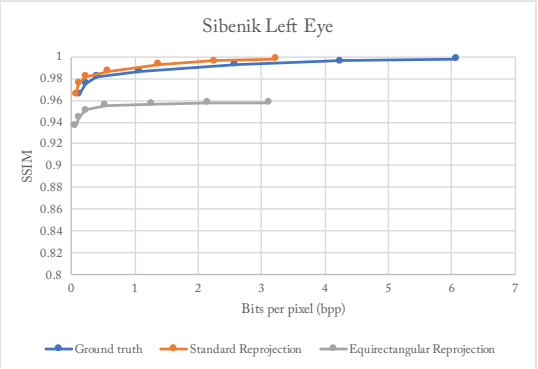
(d) Rungholt



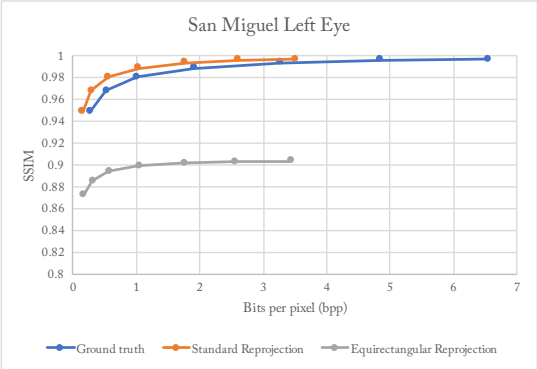
(e) Kitchen

Figure 8.8 – Rate-distortion graphs for LDR streaming of both eyes (averaged) over three methods (PSNR).

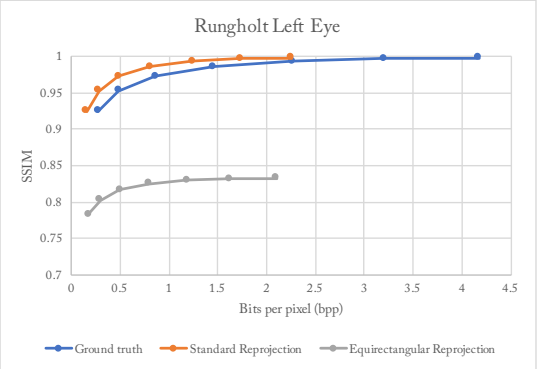
(a) Sponza



(b) Sibenik



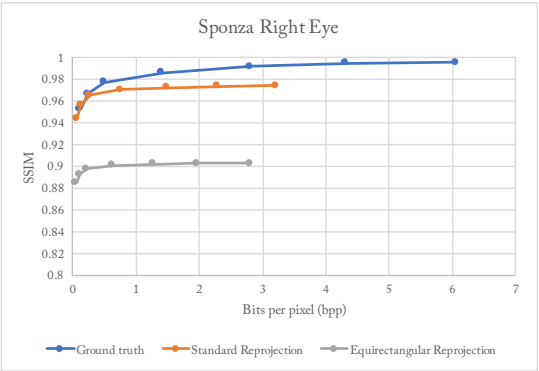
(c) San Miguel



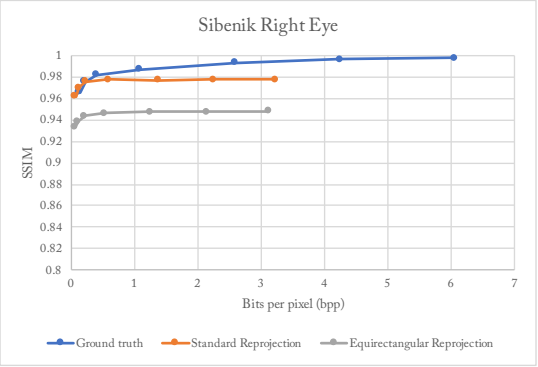
(d) Rungholt

(e) Kitchen

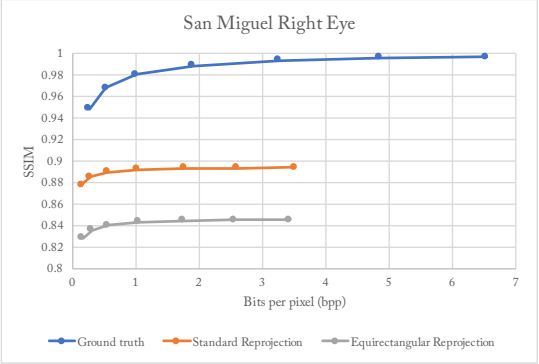
Figure 8.9 – Rate-distortion graphs for LDR streaming of the left eye over three methods (SSIM).



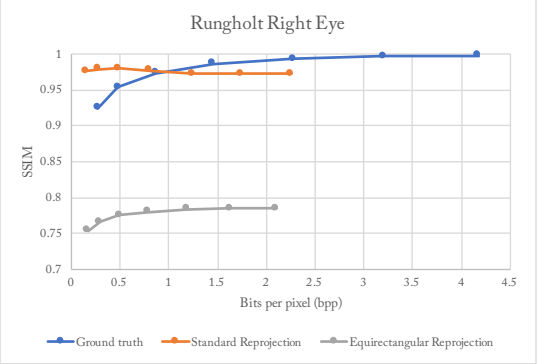
(a) Sponza



(b) Sibenik



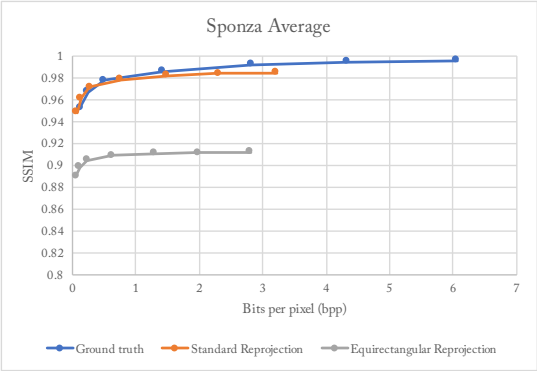
(c) San Miguel



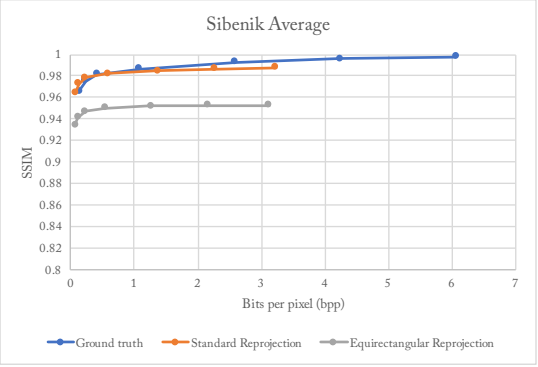
(d) Rungholt

(e) Kitchen

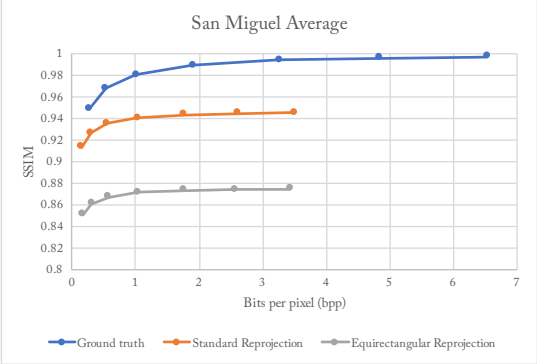
Figure 8.10 – Rate-distortion graphs for LDR streaming of the right eye over three methods (SSIM).



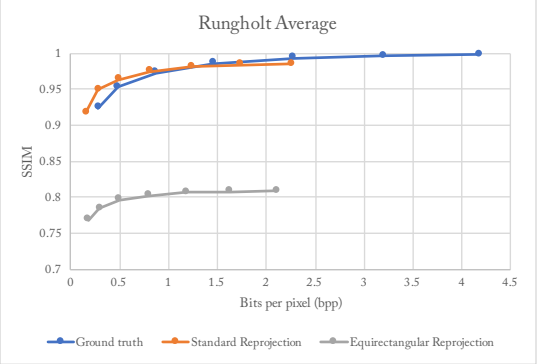
(a) Sponza



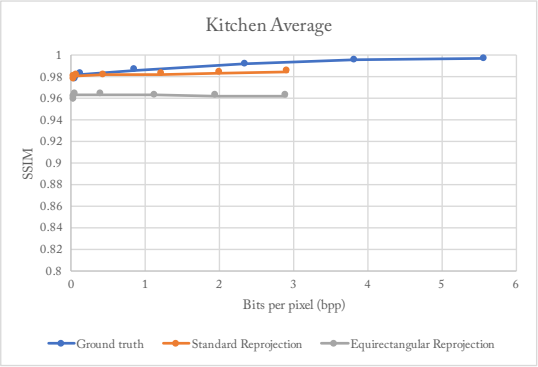
(b) Sibenik



(c) San Miguel



(d) Rungholt



(e) Kitchen

Figure 8.11 – Rate-distortion graphs for LDR streaming of both eyes (averaged) over three methods (SSIM).

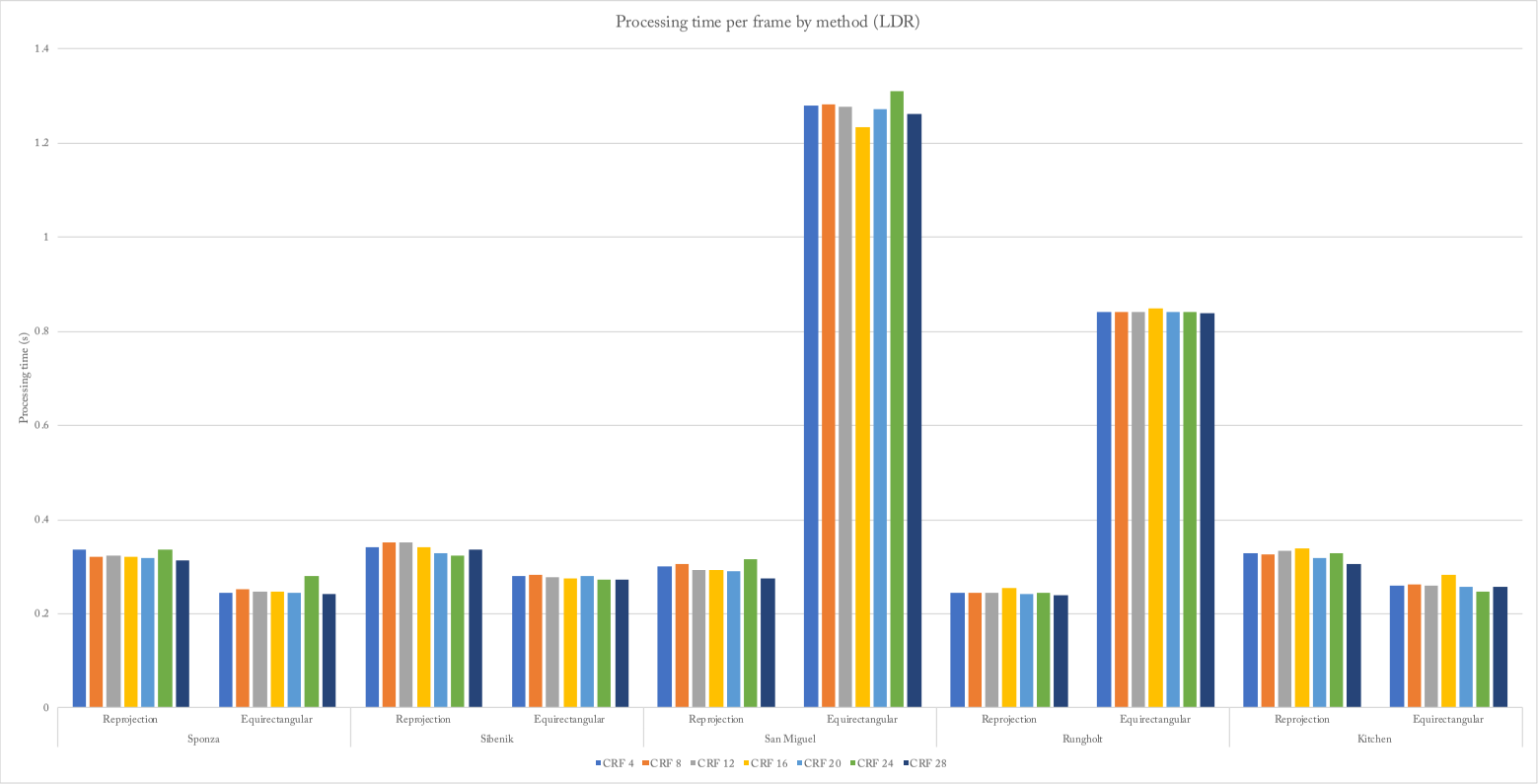
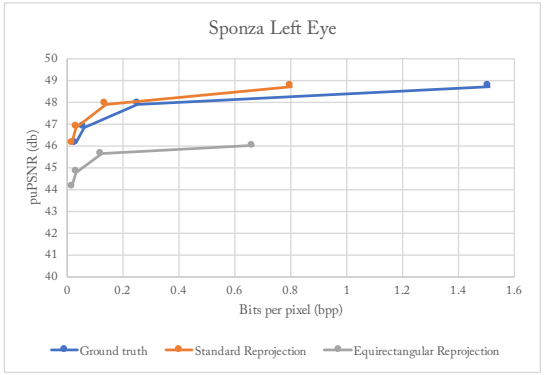
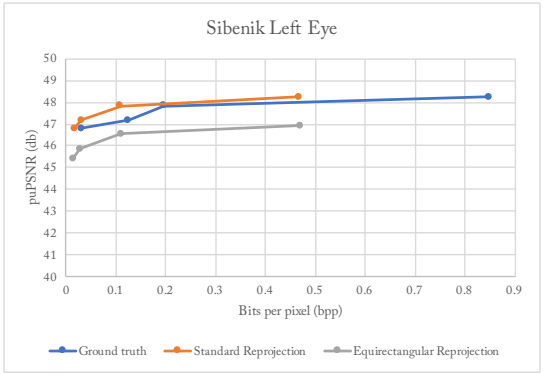


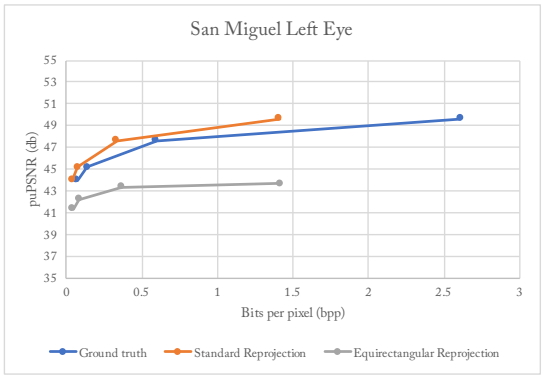
Figure 8.12 – Graph comparing processing time per frame by method (LDR). The cost of decoding video is assumed to be negligible due to the widespread availability of hardware decoders, so no processing time is allocated to the ground truth.



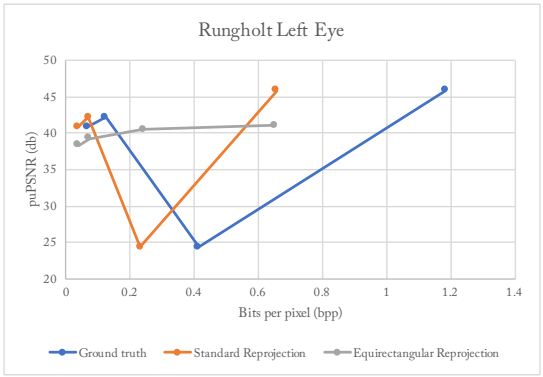
(a) Sponza



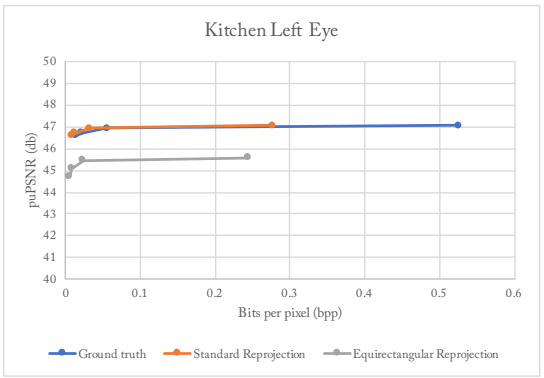
(b) Sibenik



(c) San Miguel



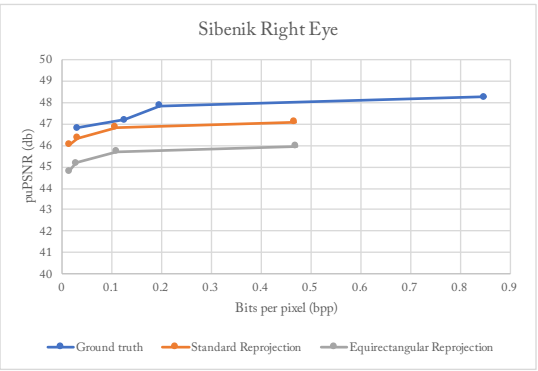
(d) Rungholt



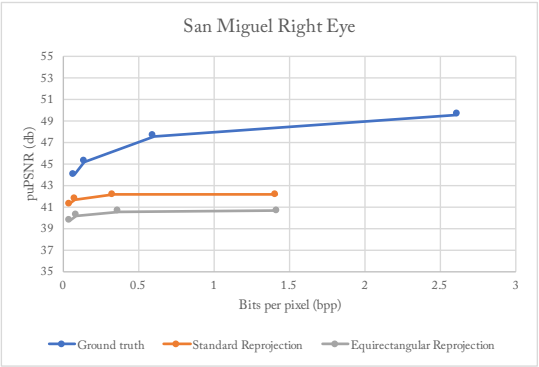
(e) Kitchen

Figure 8.13 – Rate-distortion graphs for HDR streaming of the left eye over three methods (puPSNR).

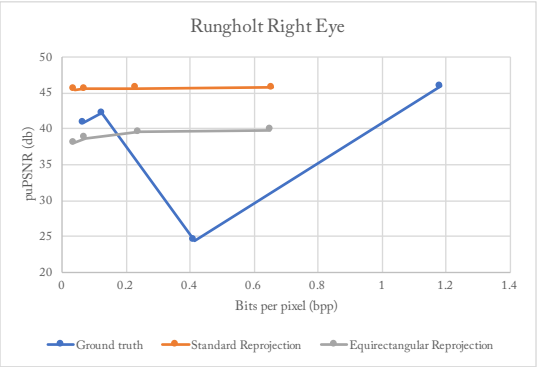
(a) Sponza



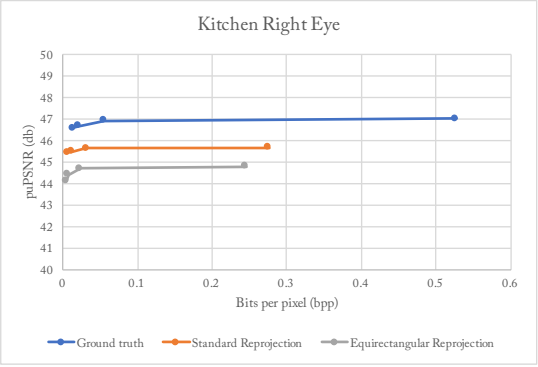
(b) Sibenik



(c) San Miguel

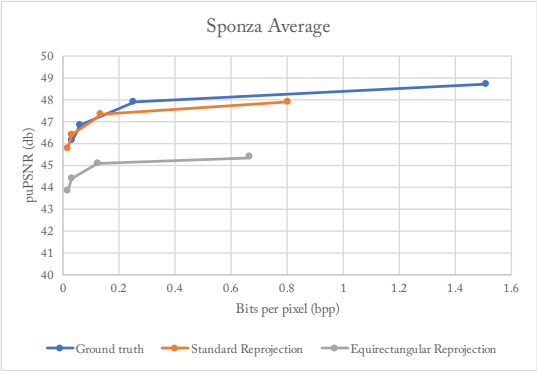


(d) Rungholt

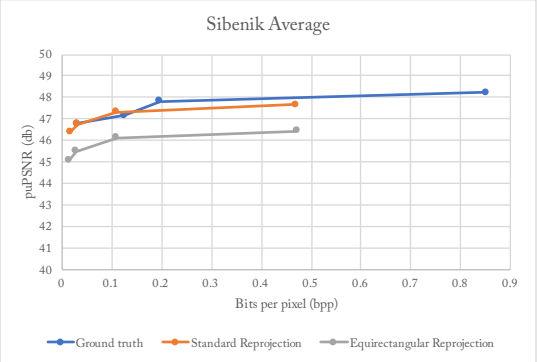


(e) Kitchen

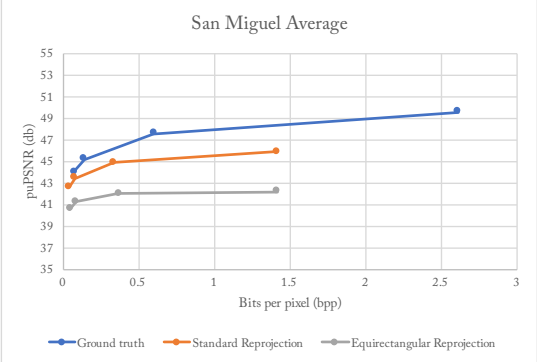
Figure 8.14 – Rate-distortion graphs for HDR streaming of the right eye over three methods (puPSNR).



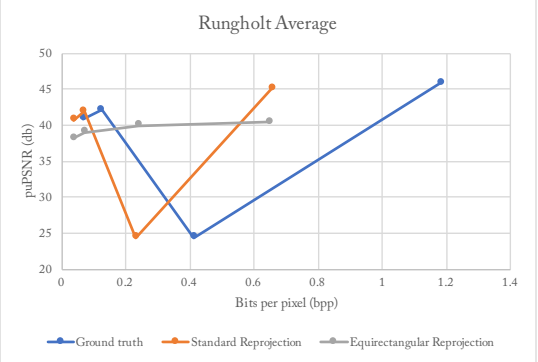
(a) Sponza



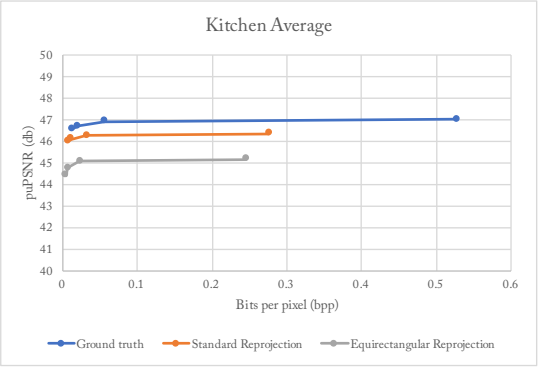
(b) Sibenik



(c) San Miguel



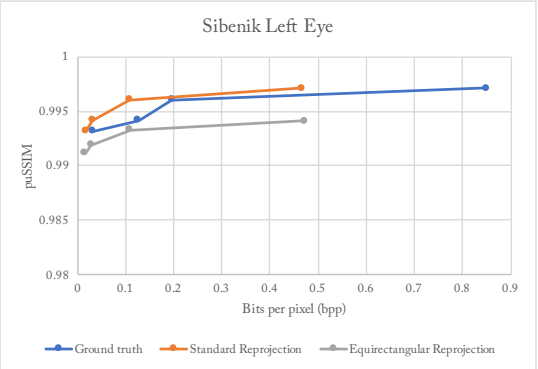
(d) Rungholt



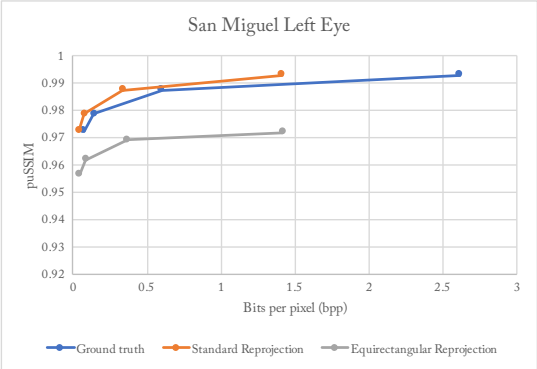
(e) Kitchen

Figure 8.15 – Rate-distortion graphs for HDR streaming of both eyes (averaged) over three methods (puPSNR).

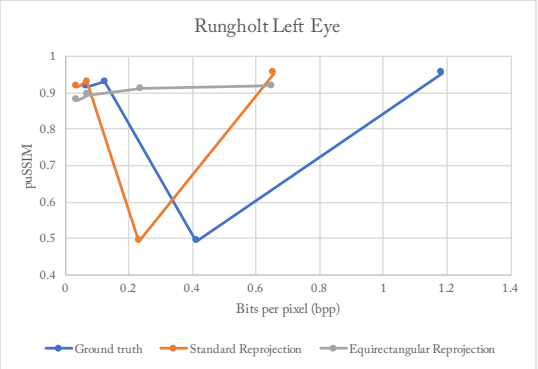
(a) Sponza



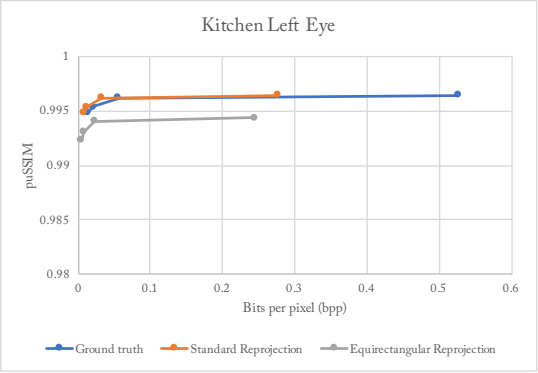
(b) Sibenik



(c) San Miguel

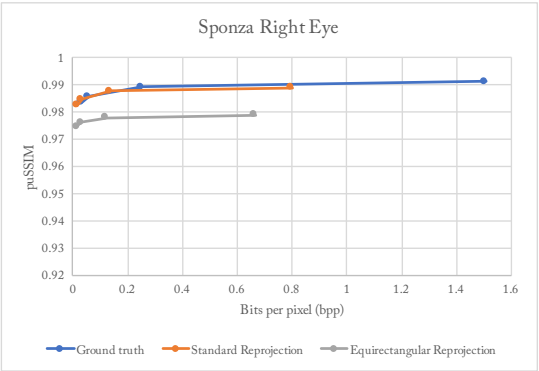


(d) Rungholt

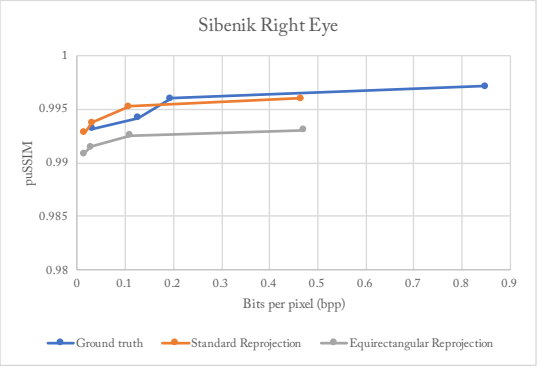


(e) Kitchen

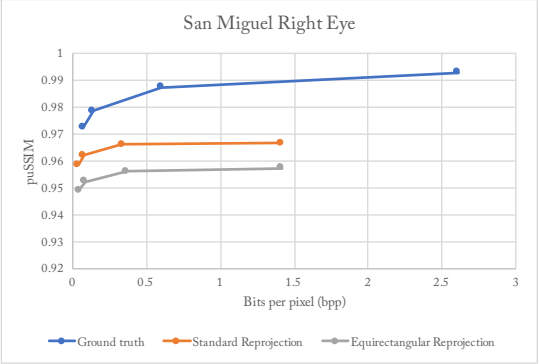
Figure 8.16 – Rate-distortion graphs for HDR streaming of the left eye over three methods (puSSIM).



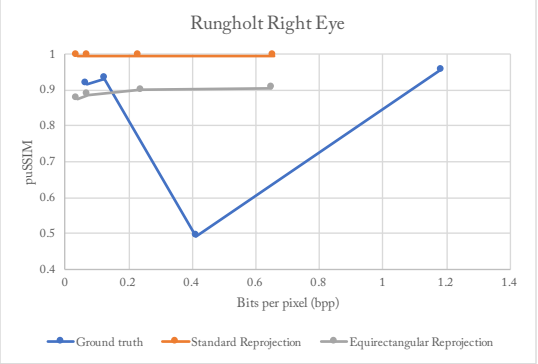
(a) Sponza



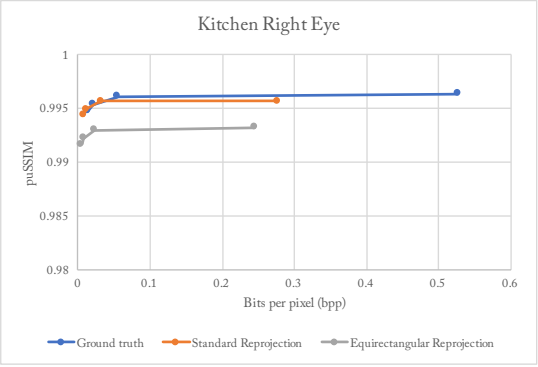
(b) Sibenik



(c) San Miguel

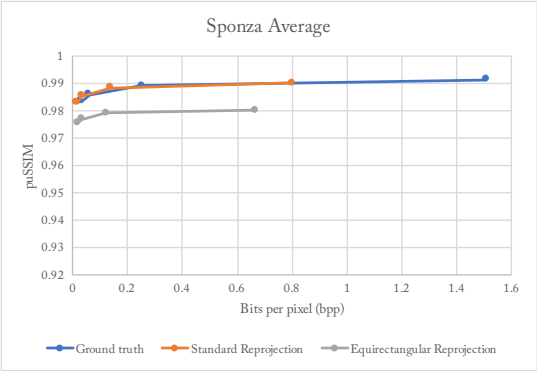


(d) Rungholt

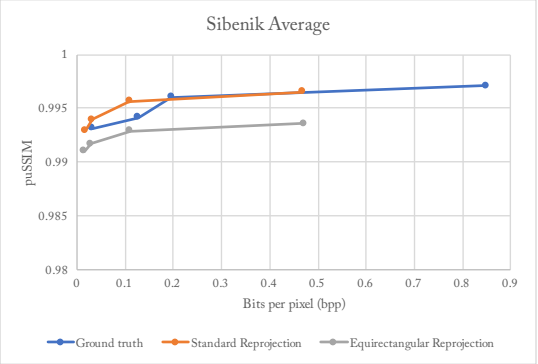


(e) Kitchen

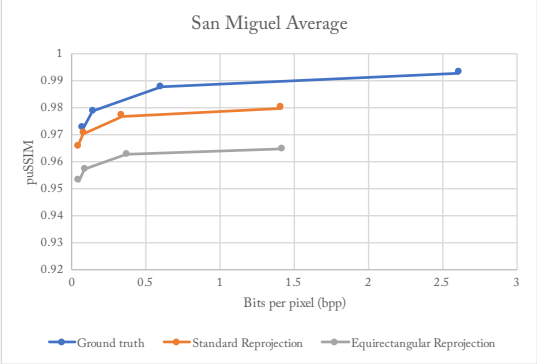
Figure 8.17 – Rate-distortion graphs for HDR streaming of the right eye over three methods (puSSIM).



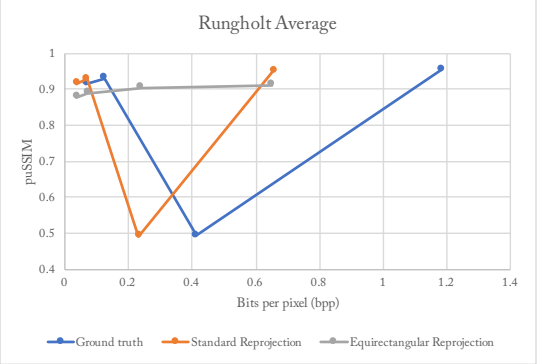
(a) Sponza



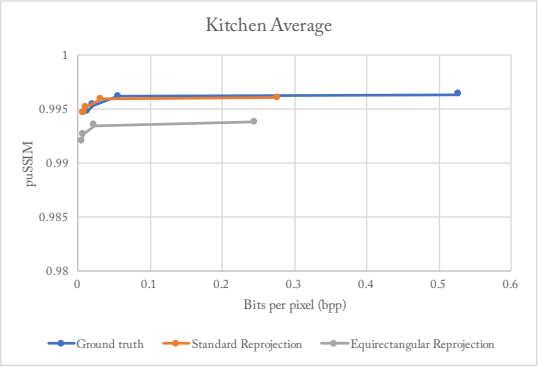
(b) Sibenik



(c) San Miguel



(d) Rungholt



(e) Kitchen

Figure 8.18 – Rate-distortion graphs for HDR streaming of both eyes (averaged) over three methods (puSSIM).

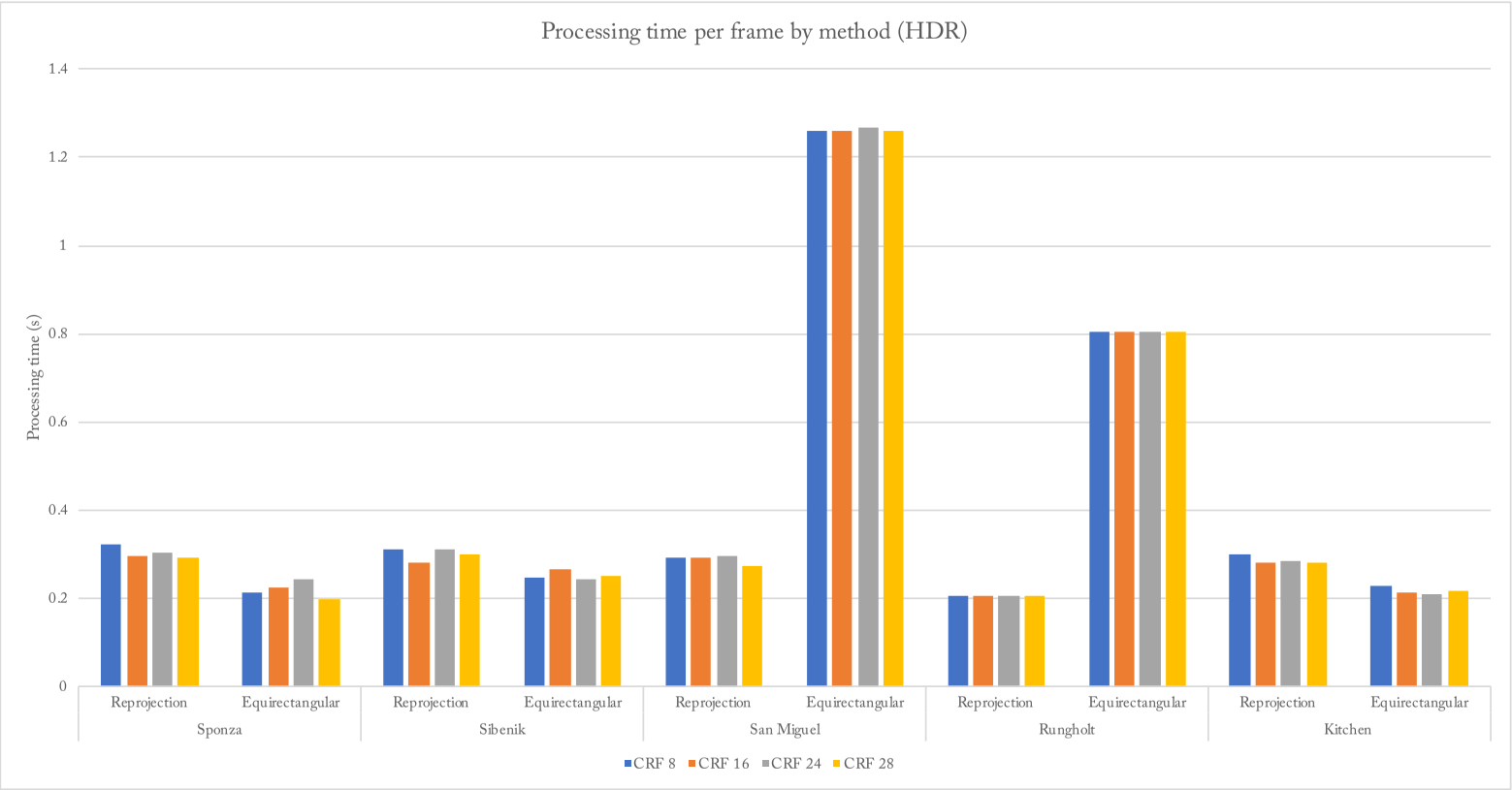


Figure 8.19 – Graph comparing processing time per frame by method (HDR). The cost of decoding video is assumed to be negligible due to the widespread availability of hardware decoders, so no processing time is allocated to the ground truth

tracing, show that for LDR footage there is a negative effect on quality as the method of transmission passes through layers of transformation. For the HDR footage, where the image is more accurately passed through transmission, the effect is lessened.

There is potential for the technique of locally-generated depth map reprojection to greatly reduce the cost of resource-intensive high-fidelity remote rendering to HMDs. Equirectangular reprojection has clear benefits for streaming to HMDs, but the cost of server-side image production would have to be increased if client-side quality is to outperform standard reprojection.

Chapter 9

Conclusion

> *Our Observation employ'd either about external, sensible Objects; or about the internal Operations of our Minds, perceived and reflected on by ourselves, is that which supplies our Understandings with all the material of thinking. These two are the Fountains of Knowledge, from whence all the Ideas we have, or can naturally have, do spring.*
>
> — John Locke, *Essay Concerning Human Understanding*

Following the research plan described in Chapter 5, this thesis has investigated through Chapter 6, Chapter 7 and Chapter 8 the streaming of a range of future-looking techniques for high-fidelity imaging, HDR video to remote rendering to 360° video rendering for virtual reality. This chapter draws conclusions from the research and then considers the implications of this research for future work.

The areas within high-fidelity imaging considered in this thesis are linked by their nature to advancements of imaging technology, but the concept that they follow from one to the other in the way they are presented here was determined by the research and the avenues of investigation which proved fruitful. Each of the individual outcomes is considered here, before a holistic conclusion is drawn.

9.1 High dynamic range

To achieve the full potential of the change in image quality that HDR video provides, it must be possible for a wide range of the population to be able to capture and display their own HDR

video content. Chapter 6 of this thesis presented a complete end-to-end HDR pipeline using off-the-shelf hardware and demonstrated that it is possible to achieve real-time performance along this entire pipeline and deliver the full range of HDR data to any display. This pipeline is capable of processing everything from capture to post-production, archival and storage, compression, transmission, and display in a straightforward manner. This framework for further research into HDR content was put to use in Chapter 7 and Chapter 8, where it enabled the easy integration of HDR provision into the experimental systems.

The software framework for developing HDR image processing pipelines which was built for this thesis is currently being prepared for release under an open-source license. It consists of a modular interface to the OpenCL GPGPU system, and maintains images entirely in full floating-point HDR from input through to output/display. This system has already been used as the basis for multiple HDR camera systems as shown in Figure 6.1 as well as a wide variety of other image processing tasks which can be accelerated using the GPU [140]. A real-time HDR relighting system was presented using the system at the NAB show in 2016, as shown in Figure 9.1.

Figure 9.1 – Real-time HDR path tracing renderer using the framework presented in Chapter 6.

9.2 Remote rendering

Chapter 7 of this thesis presents an adaptation and evaluation of metadata-based remote rendering techniques to the streaming of HDR rendered content. Metadata-based upsampling and temporal reprojection are shown to have high potential for rendered HDR content stream-

ing. This demonstration is borne out in the following chapter, Chapter 8, where the means of streaming to a VR headset are easily augmented by the use of a HDR stream to good effect.

The techniques presented were shown to be efficient methods for the remote rendering of high-fidelity, HDR graphics, based on augmentation of a low-resolution video stream with metadata buffers, adapted to HDR video encoding.

9.3 Virtual reality

Virtual reality systems have much to gain from an apt use of equirectangular projections, and it is hoped that the work described in this thesis on utilising equirectangular projections and stereo imagery will contribute towards the future adoption of this technology.

Chapter 8 demonstrates that there is potential for equirectangular projection combined with reprojection to provide significant benefits to remote rendering systems aimed at HMDs. A cost allocation must be made, however, to offset the drop in image quality caused by packing more data into the same pixel space, and inefficiency in the projection chosen. Implemented with a powerful enough distributed rendering system, the method could provide an interactive HMD experience more responsive than previous methods which depend on a fixed perspective viewpoint. In addition, there is considerable scope for offloading bandwidth-heavy information for multiple views onto available client resources.

The exploration of the area in this thesis also reveals the scope for streaming imaging to a HMD to be adapted for HDR content, providing an immediate increase in flexibility to smart client devices which can adapt the displayed content to best match the ambient lighting [105], as well as for future HMDs which are naturally HDR-capable.

The chapter demonstrates that a system which uses locally-produced depth map reprojection to provide a stereo viewpoint can outperform a ground truth encoding of both viewpoints in efficiency. The equirectangular projection-based method was shown to have high potential, but the maximum quality achievable is limited by rendered resolution.

9.4 Answering the Research Question

Considering the research question posed in Chapter 5:

- How can we adapt our methods for capture, transmission, storage, presentation and most importantly streaming of image data to the challenges presented by incoming technologies for more detailed imaging, both on existing displays as well as in virtual reality?

In total, the work presented on the streaming of high-fidelity imaging in this thesis contributes to answer this question in which the challenges of these resource-intensive incoming technologies are ameliorated by intelligent, well-adapted techniques for minimising latency, bandwidth usage, and client processing to permit the distribution of processing power where it is most needed.

9.5 Contributions

The contributions to knowledge in this thesis are:

- In Chapter 6, an end-to-end pipeline for capturing, processing and displaying HDR content.
- In Chapter 7, an discussion and adaptation of remote rendering techniques to high-fidelity graphics rendering.
- In Chapter 8, a novel use of equirectangular projection and reprojective warping by means of local depth map generation to enhance a VR experience.

Overall, this thesis contributes a set of methods and approaches to high-fidelity image streaming, showing the integration of HDR video encoding across a range of techniques including a novel combination of HDR and 360° video to stream high-fidelity graphics in real-time to a HMD.

9.6 Future work

There are a number of areas for future research:

9.6.1 HDR

The framework for processing HDR video described in this thesis is being prepared for open source release. With adaptation to process audio, it is hoped that it will form the basis of a high fidelity open-source research software for high-fidelity video processing.

Concerning the camera systems described in Chapter 6, one of the key challenges still facing HDR video capture using multiple exposures on commodity cameras is ghosting. This is especially a problem with mobile devices where the camera and the scene may both be moving. Future work is still necessary to investigate improved real-time deghosting algorithms to provide even better quality HDR video, and the possibility of capturing such on mobile devices.

9.6.2 Remote rendering

To evaluate the particular method described in Chapter 7 of compressing the metadata buffers, a more comprehensive study could be undertaken utilising an actual real-time renderer, where the path the camera takes through the scene is not precomputed. This would enable the system to be tested using real-time human inputs, requiring a stringent responsiveness in the system to avoid causing illness [9].

Similarly, testing the proposed continuous reproduction of previous frames to maintain continuity of image could be performed through subjective evaluation for preference, rather than objective comparison to a fixed set.

9.6.3 Virtual reality

Immediate further research into this area could serve to establish a direct relation between the resolution and/or bitrate required to attain parity of quality between a 360° projection and a regular perspective view. With this established, a study could be undertaken in which 360° rendering could be compared directly with the processing cost and latency of rendering two full stereo images based on a subjective participant's view motions.

Going forward, for virtual reality the most effective continuation of this work would be an attempt at adapting a standard video codec specifically for use in an equal-distortion 360° mapping. Adaptations of a spherical image to a rectangular codec will always be deficient in some regard due to the mapping, and a full 360°-oriented video codec could optimise both

rotationally and across the current seams of a rectangular mapping.

9.6.4 Future research questions

A potential future research question, covering these areas, could be phrased as follows:

- How can the technologies used for capture, transmission and storage of high-fidelity imaging be most efficiently integrated together?

9.7 Final remarks

The field of high-fidelity imaging is constantly advancing, and with it the range of technologies which must be ordered and accounted for in pursuit of it. Every further enhancement places a greater stress on the resources which must be allocated to achieve this goal, and as this process continues the importance of fast, efficient, effective utilisation will come ever closer to the fore.

The work contained within this thesis is driving towards that goal, of achieving lower demands on resources, faster responses by a virtual environment to stimulus over a distance, and the expansion of remote computing and cloud computing into areas which are still beset by challenges.

References

[1] IU SANDVINE. Global internet phenomena report. 2014. *URL: https://www. sandvine. com/downloads/general/global-internet-phenomena/2012/1h-2012-global-internet-phenomena-report. pdf*, 2010.

[2] Valve Corporation. Steam in-home streaming.

[3] Sony Interactive Entertainment Inc. Ps4 remote play, .

[4] Francesco Banterle, Alessandro Artusi, Kurt Debattista, and Alan Chalmers. *Advanced High Dynamic Range Imaging: Theory and Practice*. A. K. Peters, Ltd., Natick, MA, USA, 1st edition, 2011. ISBN 9781568817194.

[5] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0125852630.

[6] S. Kumar. Fundamental Limits to Moore's Law. *ArXiv e-prints*, November 2015.

[7] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[8] David M. Hoffman, Ahna R. Girshick, Kurt Akeley, and Martin S. Banks. Vergence–accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision*, 8(3):33, 2008. doi: 10.1167/8.3.33. URL `+http://dx.doi.org/10.1167/8.3.33`.

[9] Paul DiZio and James R. Lackner. Spatial orientation, adaptation, and motion sickness in real and virtual environments. *Presence: Teleoperators and Virtual Environments*, 1(3): 319–328, 2017/03/13 1992. URL `http://dx.doi.org/10.1162/pres.1992.1.3.319`.

[10] NVIDIA. Video capture, encoding, and streaming in a multi-gpu system. Technical report, NVIDIA, 2010.

[11] Josh McNamee, Kurt Debattista, and Alan Chalmers. Efficient Remote Rendering Using Equirectangular Projection. In Tao Ruan Wan and Franck Vidal, editors, *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2017. ISBN 978-3-03868-050-5. doi: 10.2312/cgvc.20171278.

[12] J. McNamee, K. Debattista, and A. Chalmers. Remote rendering of high dynamic range graphics content. In *2016 Digital Media Industry Academic Forum (DMIAF)*, pages 6–10, July 2016. doi: 10.1109/DMIAF.2016.7574891.

[13] Alan Chalmers, Joshua McNamee, Jonathan Hatchett, Ratnajit Mukherjee, Igor Olaizola, and Kurt Debattista. 12 bits is simply not enough for hdr video! *BEC, NAB*, 2015.

[14] Joshua McNamee, Jonathan Hatchett, Kurt Debattista, and Alan Chalmers. Live hdr video streaming on commodity hardware. volume 9599, pages 9599 – 9599 – 11, 2015. doi: 10.1117/12.2187457. URL `http://dx.doi.org/10.1117/12.2187457`.

[15] Joshua McNamee, Jon Hatchett, Kurt Debattista, and Alan Chalmers. Real time delivery of hdr video. *CVMP*, 2014.

[16] Adobe Inc. Adobe caslon pro (typeface), .

[17] Plato. *Cratylus*. 360 B.C.E.

[18] Oculus VR LLC. Oculus rift press kit, 9 2017. URL `https://www.oculus.com/press-kit/hardware/`.

[19] René Descartes. *La Dioptrique*. 1637.

[20] Philip Dutre, Kavita Bala, Philippe Bekaert, and Peter Shirley. *Advanced Global Illumination*. AK Peters Ltd, 2006. ISBN 1568813074.

[21] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM. ISBN 0-89791-196-2. doi: 10.1145/15922.15902. URL `http://doi.acm.org/10.1145/15922.15902`.

[22] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric Considerations and Nomenclature for Reflectance. *National Bureau of Standards*, 1977.

[23] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition, 2009. ISBN 1568814690, 9781568814698.

[24] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM. doi: 10.1145/1468075.1468082. URL `http://doi.acm.org/10.1145/1468075.1468082`.

[25] Solomon Boulos. Notes on efficient ray tracing. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM. doi: 10.1145/1198555.1198749. URL `http://doi.acm.org/10.1145/1198555.1198749`.

[26] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980. ISSN 0001-0782. doi: 10.1145/358876.358882. URL `http://doi.acm.org/10.1145/358876.358882`.

[27] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1999. ISBN 0201604582.

[28] Tomas Akenine-Moller, Tomas Moller, and Eric Haines. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2002. ISBN 1568811829.

[29] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500. URL http://doi.acm.org/10.1145/1365490.1365500.

[30] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *IEEE Des. Test*, 12(3):66–73, May 2010. ISSN 0740-7475. doi: 10.1109/MCSE.2010.69. URL http://dx.doi.org/10.1109/MCSE.2010.69.

[31] Ade Miller and Kate Gregory. *C++ AMP*. Pearson Education, 1st edition, 2012.

[32] M. Zlatuska and V. Havran. Ray Tracing on a GPU with CUDA – Comparative Study of Three Algorithms. In *Proceedings of WSCG'2010, communication papers*, pages 69–76, Feb 2010.

[33] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM. ISBN 0-89791-138-5. doi: 10.1145/800031.808590. URL http://doi.acm.org/10.1145/800031.808590.

[34] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM. doi: 10.1145/800186.810616. URL http://doi.acm.org/10.1145/800186.810616.

[35] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 839–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 81-7319-221-9. URL http://dl.acm.org/citation.cfm?id=938978.939190.

[36] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2):753–762, 2010. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2009.01645.x. URL http://dx.doi.org/10.1111/j.1467-8659.2009.01645.x.

[37] Andrew Adams, Natasha Gelfand, Jennifer Dolson, and Marc Levoy. Gaussian kd-trees for fast high-dimensional filtering. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 21:1–21:12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi: 10.1145/1576246.1531327. URL http://doi.acm.org/10.1145/1576246.1531327.

[38] Robert Herzog, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. Spatio-temporal upsampling on the gpu. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, Washington, DC, USA, feb 2010. ACM, ACM Press.

[39] Dawid Pająk, Robert Herzog, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. Scalable remote rendering with depth and motion-flow augmented streaming. *Computer Graphics Forum*, 30(2), 2011. Proceedings Eurographics 2011.

[40] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276497. URL http://doi.acm.org/10.1145/1276377.1276497.

[41] Lei Yang, Pedro V. Sander, and Jason Lawrence. Geometry-aware framebuffer level of detail. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1183–1188, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association. doi: 10.1111/j.1467-8659.2008.01256.x. URL http://dx.doi.org/10.1111/j.1467-8659.2008.01256.x.

[42] Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. Accelerating real-time shading with reverse reprojection caching. In *Proceedings of the 22Nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '07, pages 25–35, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 978-1-59593-625-7. URL http://dl.acm.org/citation.cfm?id=1280094.1280098.

[43] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V. Sander, and Diego Nehab. An improved shading cache for modern gpus. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '08, pages 95–101,

Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association. ISBN 978-3-905674-09-5. URL http://dl.acm.org/citation.cfm?id=1413957.1413972.

[44] Fabrizio Pece, Jan Kautz, and Tim Weyrich. Adapting standard video codecs for depth streaming. In *Proceedings of the 17th Eurographics Conference on Virtual Environments & Third Joint Virtual Reality*, EGVE - JVRC'11, pages 59–66, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association. ISBN 978-3-905674-33-0. doi: 10.2312/EGVE/JVRC11/059-066. URL http://dx.doi.org/10.2312/EGVE/JVRC11/059-066.

[45] Karsten Mueller, Philipp Merkle, Aljoscha Smolic, and Thomas Wiegand. Multiview coding using avc. In *MPEG Meeting-ISO/IEC JTC1/SC29/WG11, Bangkok, Thailand, MPEG06 M*, volume 12945, 2006.

[46] J. Ardouin, A. Lécuyer, M. Marchal, and E. Marchand. Stereoscopic rendering of virtual environments with wide field-of-views up to 360. In *2014 IEEE Virtual Reality (VR)*, pages 3–8, March 2014.

[47] Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V. Sander, Michael Wimmer, and Elmar Eisemann. Temporal coherence methods in real-time rendering. *Computer Graphics Forum*, 31(8):2378–2408, 2012. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2012.03075.x. URL http://dx.doi.org/10.1111/j.1467-8659.2012.03075.x.

[48] Isaac Newton. *Opticks*. William Innys, 4th edition, 1704.

[49] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 39–46, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218398. URL http://doi.acm.org/10.1145/218380.218398.

[50] ITU-T. H.264 : Advanced video coding for generic audiovisual services h.264 : Advanced video coding for generic audiovisual services. Technical report, ITU-T Recommendation H.264, 2014.

[51] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transfom. *IEEE Trans. Comput.*, 23(1):90–93, January 1974. ISSN 0018-9340. doi: 10.1109/T-C.1974.223784. URL `http://dx.doi.org/10.1109/T-C.1974.223784`.

[52] ITU-T. Digital compression and coding of continuous-tone still images. Technical report, ITU-T Recommendation T.81, 1992.

[53] Recommendation H ITU-T. 265 (04/13). *Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services–Coding of Moving Video, High Efficiency Video Coding, Online: http://www. itu. int/rec/T-REC-H*, pages 265–201304.

[54] Av1 bitstream and decoding process specification. Technical report, Alliance for Open Media, 2018. URL `https://aomediacodec.github.io/av1-spec/`.

[55] JTC 1 Information technology. Iso/iec 14496-14. Technical report, ISO/IEC, 2003.

[56] D.A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sept 1952. ISSN 0096-8390. doi: 10.1109/JRPROC.1952.273898.

[57] Dan S. Wallach, Sharma Kunapalli, and Michael F. Cohen. Accelerated mpeg compression of dynamic polygonal scenes. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 193–196, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0. doi: 10.1145/192161.192198. URL `http://doi.acm.org/10.1145/192161.192198`.

[58] Mark R. Bolin and Gary W. Meyer. A frequency based ray tracer. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 409–418, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218497. URL `http://doi.acm.org/10.1145/218380.218497`.

[59] Robert Herzog, Shin ichi Kinuwaki, Karol Myszkowski, and Hans-Peter Seidel. Render2mpeg: A perception-based framework towards integrating rendering and video compression. *Comput. Graph. Forum*, pages 183–192, 2008.

[60] Marc Levoy. Polygon-assisted jpeg and mpeg compression of synthetic images. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 21–28, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218392. URL http://doi.acm.org/10.1145/218380.218392.

[61] ITU-T. Information technology – lossy/lossless coding of bi-level images. Technical report, ITU-T Recommendation T.88, 2000.

[62] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, June 1987. ISSN 0001-0782. doi: 10.1145/214762.214771. URL http://doi.acm.org/10.1145/214762.214771.

[63] Alan W. Paeth. Image file compression made easy. In James Arvo, editor, *Graphics Gems II*, pages 93–100. Academic Press, 1991.

[64] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 43–54, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4. doi: 10.1145/237170.237200. URL http://doi.acm.org/10.1145/237170.237200.

[65] Cyril Crassin et al. Cloudlight: A system for amortizing indirect lighting in real-time rendering. Technical report, NVIDIA Technical Report, NVR-2013-001, 2013.

[66] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, volume 30, pages 1921–1930. Wiley Online Library, 2011.

[67] Jason Mitchell, Gary McTaggart, and Chris Green. Shading in valve's source engine. In *ACM SIGGRAPH 2006 Courses*, pages 129–142. ACM, 2006.

[68] Michael Mara, Morgan McGuire, and David Luebke. Toward practical real-time photon mapping: Efficient gpu density estimation. In *Interactive 3D Graphics and*

Games 2013, March 2013. URL http://graphics.cs.williams.edu/papers/PhotonI3D13/.

[69] Piotr Didyk, Tobias Ritschel, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. Adaptive image-space stereo view synthesis. In *Vision, Modeling and Visualization Workshop*, pages 299–306, Siegen, Germany, 2010.

[70] Immanuel Kant. *Critique of Judgement*. 1790.

[71] Helge Seetzen, Wolfgang Heidrich, Wolfgang Stuerzlinger, Greg Ward, Lorne Whitehead, Matthew Trentacoste, Abhijeet Ghosh, and Andrejs Vorozcovs. High dynamic range display systems. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 760–768. ACM, 2004.

[72] Alexa I Ruppertsberg, Marina Bloj, Francesco Banterle, and Alan Chalmers. Displaying colourimetrically calibrated images on a high dynamic range display. *Journal of Visual Communication and Image Representation*, 18(5):429–438, 2007.

[73] Samsung. Js9800, 9 2016. URL http://www.samsung.com/hk_en/tvs/suhd-js9800/UA65JS9800JXZK/.

[74] SIM2 Multimedia S.p.A. Sim2 high dynamic range display series. http://www.sim2hdr.com/, July 2015.

[75] Canon. In-camera hdr, 2012. URL http://cpn.canon-europe.com/content/education/infobank/digital_camera_features/in_camera_hdr.do.

[76] Francesco Banterle, Patrick Ledda, Kurt Debattista, and Alan Chalmers. Inverse tone mapping. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 349–356. ACM, 2006.

[77] Rafał Mantiuk, Grzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. Perception-motivated high dynamic range video encoding. *ACM Trans. Graph.*, 23(3): 733–741, 2004. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1015706.1015794.

[78] F. Banterle, A. Artusi, K. Debattista, P. Ledda, G. Bonnet, G.J. Edwards, and A. Chalmers. Video data compression, September 29 2011. URL https://www.google.com/patents/US20110235720. US Patent App. 12/984,992.

[79] Greg Ward. Real pixels. *Graphics Gems II*, pages 80–83, 1991.

[80] Gregory Ward Larson. Logluv encoding for full-gamut, high-dynamic range images. *J. Graph. Tools*, 3(1):15–31, March 1998. ISSN 1086-7651. doi: 10.1080/10867651.1998.10487485. URL http://dx.doi.org/10.1080/10867651.1998.10487485.

[81] Industrial Light & Magic. OpenEXR. http://www.openexr.com/, October 2014.

[82] Alexandre Benoit, David Alleysson, Jeanny Herault, and Patrick Le Callet. Spatio-temporal tone mapping operator based on a retina model. In *Computational Color Imaging*, pages 12–22. Springer, 2009.

[83] Ronan Boitard, Kadi Bouatouch, Remi Cozot, Dominique Thoreau, and Adrien Gruson. Temporal coherency for video tone mapping. In *SPIE Optical Engineering+ Applications*, pages 84990D–84990D. International Society for Optics and Photonics, 2012.

[84] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: 10.1145/258734.258884. URL http://dx.doi.org/10.1145/258734.258884.

[85] Alan Chalmers, Gerhard Bonnet, Francesco Banterle, Piotr Dubla, Kurt Debattista, Alessandro Artusi, and Christopher Moir. High-dynamic-range video solution. In *ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation*, SIGGRAPH ASIA '09, pages 71–71, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-878-0. doi: 10.1145/1665137.1665190. URL http://doi.acm.org/10.1145/1665137.1665190.

[86] Michael D Tocci, Chris Kiser, Nora Tocci, and Pradeep Sen. A versatile hdr video

production system. In *ACM Transactions on Graphics (TOG)*, volume 30, page 41. ACM, 2011.

[87] Brian Karr, Alan Chalmers, and Kurt Debattista. High dynamic range digital imaging of spacecraft. In F. Dufaux, P. Le Callet, R. Mantiuk, and M. Mrak, editors, *High Dynamic Range - From Acquisition to Display and Applications*. 2015. Forthcoming.

[88] MIT. Halide. `http://halide-lang.org/`, July 2015.

[89] Ajit Motra and Herbert Thoma. An adaptive logluv transform for high dynamic range video compression. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2061–2064. IEEE, 2010.

[90] Yang Zhang, Erik Reinhard, and David Bull. Perception-based high dynamic range video compression with optimal bit-depth transformation. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 1321–1324. IEEE, 2011.

[91] Scott Miller, Mahdi Nezamabadi, and Scott Daly. Perceptual signal coding for more efficient usage of bit codes. In *SMPTE Conferences*, volume 2012, pages 1–9. Society of Motion Picture and Television Engineers, 2012.

[92] Tim Borer. Non-linear opto-electrical transfer functions for high dynamic range television. 2014.

[93] Rafał Mantiuk, Alexander Efremov, Karol Myszkowski, and Hans-Peter Seidel. Backward compatible high dynamic range mpeg video compression. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 713–723. ACM, 2006.

[94] Greg Ward and Maryann Simmons. Jpeg-hdr: A backwards-compatible, high dynamic range extension to jpeg. In *ACM SIGGRAPH 2006 Courses*, page 3. ACM, 2006.

[95] Chul Lee and Chang-Su Kim. Rate-distortion optimized compression of high dynamic range videos. In *Signal Processing Conference, 2008 16th European*, pages 1–5. IEEE, 2008.

[96] F. Banterle, A. Artusi, K. Debattista, P. Ledda, A. Chalmers, G.J. Edwards, and G. Bonnet. Hdr video data compression devices and methods, June 27 2008. URL `https://www.google.com/patents/EP2144444B1?cl=en`. EP Patent 2,144,444.

[97] Gary J Sullivan, J-R Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[98] High dynamic range electro-optical transfer function of mastering reference displays. *SMPTE ST 2084:2014*, pages 1–14, Aug 2014. doi: 10.5594/SMPTE.ST2084.2014.

[99] Association of Radio Industries and Businesses (ARIB). STD-B67: Essential parameter values for the extended image dynamic range television (EIDRTV) system for programme production, July 2015.

[100] J. Hatchett, K. Debattista, R. Mukherjee, T. B. Rogers, and A Chalmers. Highly efficient hdr video compression. Technical report, JCT-VC input document, JCTVC-W0072, San Diego, February 2016.

[101] George Berkeley. An essay towards a new theory of vision. 1732.

[102] Gary J Sullivan, J-R Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[103] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje. The latest open-source video codec vp9 - an overview and preliminary results. In *2013 Picture Coding Symposium (PCS)*, pages 390–393, Dec 2013. doi: 10.1109/PCS.2013.6737765.

[104] MulticoreWare Inc. x265 hevc encoder / h.265 video codec, .

[105] M. Melo, M. Bessa, K. Debattista, and A. Chalmers. Evaluation of tone-mapping operators for hdr video under different ambient luminance levels. *Computer Graphics Forum*, 34(8):38–49, 2015. ISSN 1467-8659. doi: 10.1111/cgf.12606. URL `http://dx.doi.org/10.1111/cgf.12606`.

[106] B. de Spinoza and E.M. Curley. *Ethics*. Penguin classics. Penguin Books, 1996. URL https://books.google.co.uk/books?id=YUjuAAAAMAAJ.

[107] E. Reinhard, E. Francois, R. Boitard, C. Chamaret, C. Serre, and T. Pouli. High dynamic range video production, delivery and rendering. *SMPTE Motion Imaging Journal*, 124(4):1–8, May 2015. ISSN 1545-0279. doi: 10.5594/j18552.

[108] Benjamin Guthier, Stephan Kopf, and Wolfgang Effelsberg. A real-time system for capturing hdr videos. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 1473–1476, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1089-5. doi: 10.1145/2393347.2396524. URL http://doi.acm.org/10.1145/2393347.2396524.

[109] Magic lantern. http://www.magiclantern.fm/, July 2015.

[110] R. Ramírez Orozco, I. Martin, C. Loscos, and P.-P. Vasquez. Full high-dynamic range images for dynamic scenes. volume 8436, pages 843609–843609–16, 2012. doi: 10.1117/12.922825. URL http://dx.doi.org/10.1117/12.922825.

[111] Miguel Granados, Kwang In Kim, James Tompkin, and Christian Theobalt. Automatic noise modeling for ghost-free hdr reconstruction. *ACM Trans. Graph.*, 32(6):201:1–201:10, November 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508410. URL http://doi.acm.org/10.1145/2508363.2508410.

[112] COST Action IC1005. http://ic1005-hdri.inesctec.pt/, July 2015.

[113] Ajay Luthra, Edouard Francois, and Walt Husak. Draft requirements and explorations for hdr/wcg content distribution and storage. *ISO/IEC JTC1/SC29/WG11 MPEG2014 N*, 14510, 2014.

[114] VideoLAN Organisation. x264. https://www.videolan.org/developers/x264.html, July 2015.

[115] Tuan Q Pham and Lucas J Van Vliet. Separable bilateral filtering for fast video preprocessing. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[116] Haarm-Pieter Duiker and George Borshukov. Filmic tone mapping. 2006. URL http://duikerresearch.com/2015/09/filmic-tonemapping-ea-2006/.

[117] Frédéric Drago, Karol Myszkowski, Thomas Annen, and Norishige Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. In *Computer Graphics Forum*, volume 22, pages 419–426. Wiley Online Library, 2003.

[118] Ian Hubert. Tears of steel. 2012.

[119] Sébastien Lasserre, Fabrice LeLéannec, and Edouard Francois. Description of hdr sequences proposed by technicolor. *ISO/IEC JTC1/SC29/WG11 JCTVC-P0228], IEEE, San Jose, USA*, 2013.

[120] McDonough and Antognazza. Leibniz and optics. URL http://www.oxfordhandbooks.com/10.1093/oxfordhb/9780199744725.001.0001/oxfordhb-9780199744725-e-007.

[121] Lucas Digital Ltd. LLC. *Technical Introduction to OpenEXR*, November 2006. URL http://www.openexr.com/TechnicalIntroduction.pdf.

[122] Q. Huynh-Thu and M. Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics Letters*, 44(13):800–801, June 2008. ISSN 0013-5194. doi: 10.1049/el:20080522.

[123] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13 (4):600–612, April 2004. ISSN 1057-7149. doi: 10.1109/TIP.2003.819861.

[124] Scott Daly. Digital images and human vision. chapter The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity, pages 179–206. MIT Press, Cambridge, MA, USA, 1993. ISBN 0-262-23171-9. URL http://dl.acm.org/citation.cfm?id=197765.197783.

[125] Rafał Mantiuk, Karol Myszkowski, and Hans-Peter Seidel. Visible difference predicator for high dynamic range images. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 2763–2769, 2004.

[126] Manish Narwaria, Matthieu Perreira Da Silva, and Patrick Le Callet. Hdr-vqm: An objective quality measure for high dynamic range video. *Signal Processing: Image Communication*, 35:46 – 60, 2015. ISSN 0923-5965. doi: http://dx.doi.org/10.1016/j.image.2015.04.009. URL http://www.sciencedirect.com/science/article/pii/S0923596515000703.

[127] CryTeK. http://www.crytek.com/cryengine/cryengine3/downloads.

[128] Marko Dabrovic and Mihovil Odak. Sibenik Cathedral. http://http://hdri.cgtechniques.com/~sibenik2/download, 2002.

[129] Anat Grynberg and Greg Ward. Conference model, 1990.

[130] G. Debord and K. Knabb. *Society of the Spectacle*. Rebel Press, 1983. ISBN 9780946061129. URL https://books.google.co.uk/books?id=yBB4f_dQ3rIC.

[131] YouTube. https://support.google.com/youtube/answer/6178631?hl=en-gb.

[132] Facebook. https://code.facebook.com/posts/1638767863078802/under-the-hood-building-360-video/.

[133] Unreal Engine. https://docs.unrealengine.com/latest/int/blueprintapi/input/headmounteddisplay/getori

[134] TunÇ Ozan Aydın, Rafał Mantiuk, and Hans-Peter Seidel. Extending quality metrics to full dynamic range images. In *Human Vision and Electronic Imaging XIII*, Proceedings of SPIE, pages 6806–10, San Jose, USA, January 2008.

[135] Morgan McGuire. Computer graphics archive, August 2011. URL http://graphics.cs.williams.edu/data. http://graphics.cs.williams.edu/data.

[136] Mark Segal and Kurt Akeley. Opengl 4.5 core profile. Technical report, Khronos Group Inc., 2017.

[137] Eugene Lapidous and Guofang Jiao. Optimal depth buffer for low-cost graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, HWWS '99, pages 67–73, New York, NY, USA, 1999. ACM. ISBN 1-58113-170-4. doi: 10.1145/311534.311579. URL http://doi.acm.org/10.1145/311534.311579.

[138] NVIDIA. https://developer.nvidia.com/content/depth-precision-visualized.

[139] John Locke. *An Essay Concerning Human Understanding*. 1690.

[140] Miguel Melo, Luís Barbosa, Maximino Bessa, Kurt Debattista, and Alan Chalmers. Context-aware hdr video distribution for mobile devices. *Multimedia Tools and Applications*, 76(15):16605–16623, Aug 2017. ISSN 1573-7721. doi: 10.1007/s11042-016-3940-y. URL `https://doi.org/10.1007/s11042-016-3940-y`.